VŠB – Technická univerzita Ostrava

Fakulta strojní

Katedra robotiky

Čtvernohý robot s vnitřní diagnostikou

Quadruped Robot with Inside Diagnostics

Student: Bc. Robert Pastor

Vedoucí diplomové práce: Ing. Zdenko Bobovský, PhD.

Ostrava 2017

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra robotiky

Zadání diplomové práce

Student: **Bc. Robert Pastor**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 2301T013 Robotika

Téma: Čtvernohý robot s vnitřní diagnostikou
Quadruped Robot with Inside Diagnostics

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Analýza současného stavu - přehled čtvernohých robotů, přehled systémů pro vnitřní diagnostiku
2. Specifikace požadavkového listu
3. Mechanická část - návrh mechanické části
4. Hardwarová část - hardwarové komponenty pro vnitřní diagnostiku
5. Softwarová část operátor
    - 5.1 Komunikace so simulačním programem
    - 5.2 Komunikace s robotem
6. Simulace pohybu robota - využití dostupného simulačního programu
7. Softwarová část robot
    - 7.1 Kinematika kráčení
    - 7.2 Antikolízní systém
    - 7.3 Stabilizační systém
8. Práci též doložte v elektronické podobě ve formátu MS WORD

Seznam doporučené odborné literatury:

Robotika – Servisné roboty, Navrhovanie, Konštrukcia, Riešenia. Košice: Vydavateľstvo Michala Vaška, 2008. ISBN 978-80-7165-713-2.
Robotika – Metodika nasadzovania servisných robotov. Košice: Technická univerzita v Košiciach, 2013. ISBN 978-80-5553-1523-2.
Teorie průmyslových robotů. Košice: Vienala Košice, 2000. ISBN 80-88922-35-6.

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě ....

podpis studenta

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou (bakalářskou) práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.

- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou (bakalářskou) práci užít (§ 35 odst. 3).

- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.

- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.

- bylo sjednáno, že užít své dílo – diplomovou (bakalářskou) práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě :.....

.....

podpis

Jméno a příjmení autora práce: Robert Pastor

Adresa trvalého pobytu autora práce: Gudrichova 146, Raduň 747 61, Česká republika

ANOTACE DIPLOMOVÉ PRÁCE

PASTOR, R. *Čtvernohý robot s vnitřní diagnostikou : diplomová práce.* Ostrava : VŠB – Technická univerzita Ostrava, Fakulta strojní, Katedra robotiky, 2017, 69 s. Vedoucí práce: Bobovský, Z.

Cílem této práce je navrhnout čtyřnohý robot inspirovaný bioorganismy a vybavit jej senzory a výpočty pro vnitřní diagnostiku systému. Práce se zabývá konstrukčním návrhem mechanismu, kde většina konstruovaných dílů je určena pro výrobu na 3D tiskárně. Dále se zabývá specifikací hardwarových komponentů. Pro použitý mikrokontroler byla vyrobena rozšiřující deska plošných spojů pro připojení servomotorů. Senzorický subsystém robotu obsahuje detekci doteků noh, měření proudu, napětí akumulátoru a úhlů natočení těla robotu pomocí IMU. Práce popisuje tvorbu programů na PC a v robotu, použitou strukturu kódu a použité algoritmy. Robot komunikuje pomocí vlastního sériového protokolu. V robotu probíhají výpočty generování pohybu, inverzní kinematiky, polohy těžiště a detekce kolizí. Pro robot byl vytvořen simulační model v programu V-Rep, na kterém byly testovány použité algoritmy.

ANNOTATION OF MASTER THESIS

PASTOR, R. *Quadruped Robot with Inside Diagnostics : Master Thesis.* Ostrava : VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Robotics, 2017, 69 p. Thesis head: Bobovský, Z.

The goal of this thesis is to design a quadruped robot inspired by bioorganisms and equip it with internal sensors. The thesis deals with mechanical design of the robot, where the majority of parts is designed for 3D printing. The thesis also deals with hardware specifications. An additional PCB has been manufactured to expand the capabilities of the used microcontroller. The sensory subsystem consists of foothold detection, current and voltage measurements and orientation detection using an IMU. The thesis describes development of the software for PC and the robot, used code structure and algorithms. The robot communicates using a custom serial protocol. Movement generation, inverse kinematics, center of gravity position and collision detection are all being calculated onboard the robot. A simulation model has been built in V-Rep, on which the used algorithms were tested.

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých značek a symbolů

Značka	Veličina	Jednotky
c	pomocná délka při výpočtu kinematiky	mm
l_i	Délka segmentu i	mm
m_n	Hmotnot nohy	g
m_{tr}	Hmotnost těla robotu	g
P	Poloha bodu	mm
R_i	Hodnota rezistoru	ohm
S_B	Bázový souřadnicový systém	-
S_{Ni}	Souřadnicový systém nohy i	-
S_R	Souřadnicový sytém těla robotu	-
S_{Si}	Souřadnicový systém segmentu i	-
T	Transformační matice	
T_{RN1}	Transformační matice z S_R do S_{Ni}	
U_{in}	Vstupní napětí	V
U_{out}	Výstupní napětí	V
α	pomocný úhel při výpočtu kinematiky	rad
γ	pomocný úhel při výpočtu kinematiky	rad
ε_i	Úhlová kompenzace motoru	rad
θ	Úhel natočení stupňů volnosti	rad
ρ	vzdálenost bodu P od osy Z	mm
ψ	pomocný úhel při výpočtu kinematiky	rad

Seznam použitých zkratek

Značka	Popis
FSR	force sensing resistor
GND	uzemnění
IMU	inertial measurement unit
PC	osobní počítač
PLA	polylactic acid, plast používaný v 3D tisku
RX	receiver, datový vstup mikrokontroleru
SS	souřadnicový systém
TTL	tranzistorově-tranzistorová logika
TX	transmitter, datový výstup mikrokontroleru
UART	asynchronní sériové rozhraní

0 Úvod

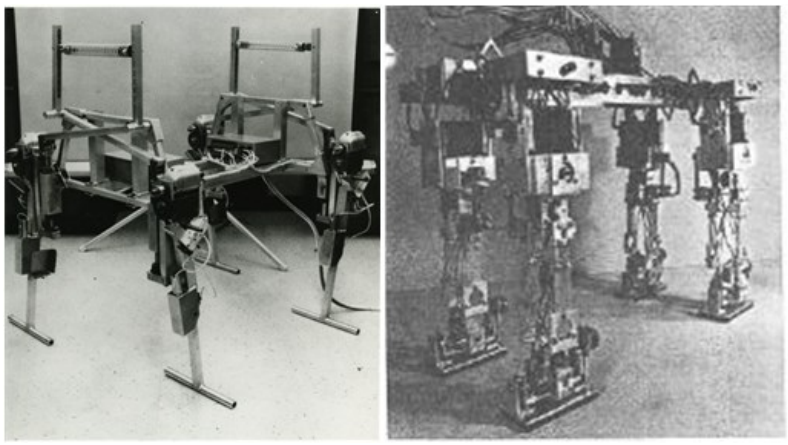
Kráčející roboty mají potenciální využití pro vykonávání činností v komplexním a nebezpečném prostředí. Například záchrana při nehodách, pohyb v podzemních dolech, atomových elektrárnách nebo pro vojenský transport. Oproti kolovým robotům, které fungují dobře na rovných površích, mohou kráčející roboty proniknout přes nerovné povrchy. To v důsledku motivovalo spoustu výzkumu v oblasti kráčejících robotů v posledních letech. I přes velké množství úspěchů v robotice kráčející roboty stále zaostávají za schopnostmi jejich biologických vzorů.

Tato práce se zabývá tvorbou kráčejícího robotu s čtyřmi nohami, který je osazen senzory monitorující vnitřní stav systému, jako je orientace robotu, odebírané proudy nebo napětí akumulátoru. Diagnostika vnitřního stavu je vyhodnocována i analyticky pomocí výpočtů aktuální polohy těžiště robotu a kontroly proti kolizím v mechanismu.

V první kapitole je uveden přehled současných čtyřnohých robotů ze zahraničních laboratoří. Druhá kapitola specifikuje seznam požadavků na vyvíjený systém. V třetí kapitole je popsána konstrukce robotu, jeho jednotlivé části, kinematika, návrh vyráběných dílů a jejich sestavení. Čtvrtá kapitola popisuje jednotlivé hardwarové komponenty, jejich vlastnosti a zapojení v robotu. V páté kapitole je tvorba simulačních modelů v programu V-Rep. Software pro operátora robotu je popsán v kapitole šest. Kapitola sedm popisuje software v robotu, jeho strukturu a použité algoritmy. V osmé kapitole jsou popsány testy fyzického robotu a další možné rozšíření systému.

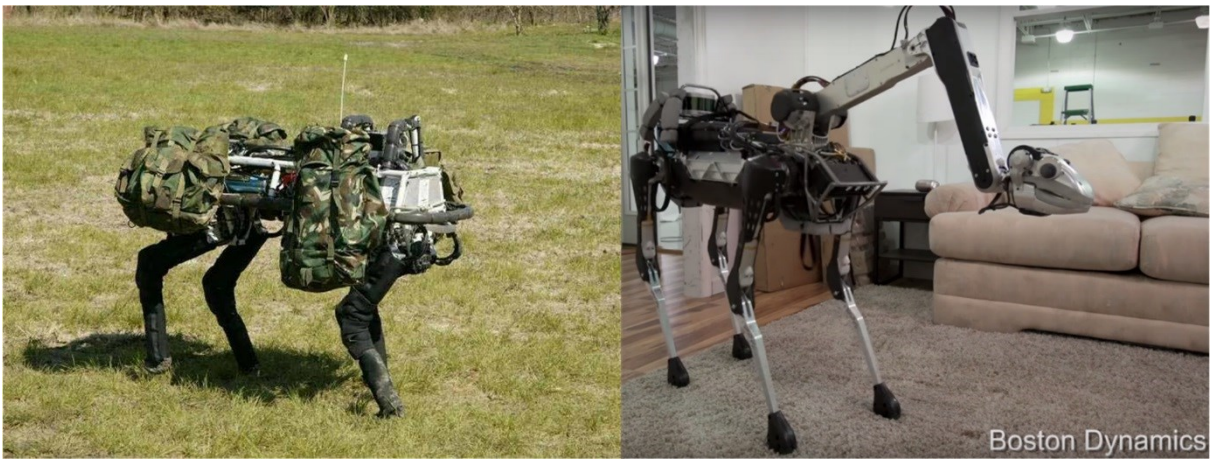
1 Přehled současného stavu řešené problematiky

Výzkum čtvernohých robotů začal v 60. letech 20. století. Jako první takový robot je možno označit čtyřnohého Phony Pony, který měl dva stupně volnosti v každé noze a byl plně řízen počítačem [1]. V 80. letech vznikla spousta kráčejících a skákajících robotů na University of Tokyo ve spolupráci s Massachusetts Institute of Technology. Mezi nimi i čtyřnohé roboty Collie-1 a Collie-2, které dokázaly dynamicky klusat a běhat [2].



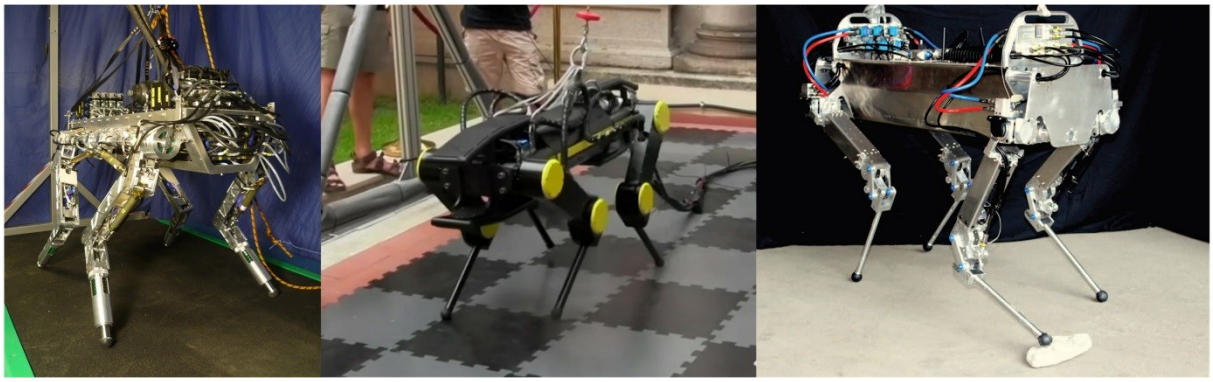
Obr. 1 Robot Phony Pony (vlevo) a Collie-1 (vpravo) [1] [2]

Jeden z nejznámějších robotů se čtyřmi nohami je BigDog vyvinutý firmou Boston Dynamics [3]. BigDog, se svou hmotností 90kg dokáže unést přes 50kg nákladu, je schopný překonat až 45 stupňové svahy a procházet lesem. Byl na něm představen stabilní pohyb i v přítomnosti náhlých vnějších sil. Robot LittleDog od Boston Dynamics zvládá přejít širokou škálu členitých povrchů s rozdílnou úrovní složitosti [4]. Boston Dynamics postupem od roku 2005 představilo několik nových modelů čtyřnohých robotů vycházejících z robotu BigDog, nejnovějším robotem je plně elektrický SpotMini vybavený manipulátorem s celkovou hmotností 25 kg a výdrží na baterii 90 minut [5].



Obr. 2 Roboty od Boston Dynamics. BogDog (vlevo) a SpotMini (vpravo) [3] [5]

Italský institut technologie vyvíjí od roku 2007 čtyřnohý robot HyQ, který kombinuje hydraulický pohon s elektrickým motorem [6]. V roce 2015 představili jeho novou verzi HyQ2Max, využívající kolenních kloubů inspirovaných biologickými šlachami. V současné době vyvíjí zmenšenou verzi MiniHyQ, s hmotností pouhých 24 kg [7].



Obr. 3HyQ, HyQ2Max a MiniHyQ [6] [8] [7]

Čtyřnohý robot SCalfl vyvinut na Shandong university ve tvaru poloelipsy zvládá klusání na měkkém povrchu [9]. Se svou hmotností 80 kg unese náklad 78 kg.

Robot s nohami tvořenými pantografickým mechanismem byl vyvinut na Shanghai Jiao Tong University [10]. Robot „Baby Elephant“ měří 1 metr a váží 130 kg. Je navržen jako mechanický nosič s nosností přes 50 kg.

Robot JINPOONG [11] je vyvíjen na Korejském institutu průmyslových technologií KITECH. Disponuje 16 aktivními stupni volnosti, které jsou poháněny hydrogenerátorem v robotu.

Obr. 4 Roboty SCalf1, Baby Elephant a JINPOONG [9] [10] [11]

Malé roboty do 20 kg si často vystačí s elektrickými pohony. U těžších robotů se osvědčily hydraulické aktuátory a větší roboty tak v sobě musí mít zabudován hydrogenerátor. V [7] jsou čtyřnohé roboty využívající hydraulický pohon porovnány v přehledné tabulce.

Tabulka 1 Přehled čtyřnohých robotů s hydraulickým pohonem [7]

Name	Mass [kg] pump off (on)	Dimensions [m^3] L [m] × W [m] × H [m]	DoF per leg	Trq. Ctrl.
SCalf	78 (123)	1.10 × 0.49 × 1.00	3	Yes
HyQ	75 (98)	1.00 × 0.50 × 1.00	3	Yes
Baby Elephant	90 (130)	1.20 × 0.60 × 1.00	3	No
BigDog	N/A (110)	1.10 × 0.40 × 1.00	4	Yes
JINPOONG	80 (120)	1.10 × 0.40 × 1.20	4	No
RLA-1	60.2 (N/A)	1.10 × 0.67 × 1.00	3	No
LS3	N/A	> BigDog	3	N/A
Wildcat	N/A	N/A	3	N/A
MiniHyQ	24 (35)	0.85 × 0.35 × 0.77	3	Yes

Inspirace živými organismy nebývá pouze v mechanických konstrukcích robotů. Po vzoru zvířat používají mnohé kráčející roboty ke svému pohybu speciální neuronové sítě zvané cental pattern generator [12] [13]. Ty jsou schopny vytvářet rytmické vzory bez potřeby senzorických informací. Uvnitř často fungují jako oscilátory tvořené navzájem se inhibujícími neurony. Přesto, že zpětná vazba senzorů není u těchto generátorů pohybu nutná, hraje důležitou roli při adaptaci rytmických vzorů pohybu, což je základem koordinace těla a generátoru pohybu. Central pattern generátory zvládají plynule přecházet mezi styly pohybu jako je chůze, klusání a běh. Jejich použití je však velmi omezené a vyžaduje specifické podmínky. Ladění těchto speciálních neuronových sítí je časově náročné a s nejistými výsledky.

Často používaným přístupem k řízení pohybu kráčejících robotů je použití analytických metod a matematických modelů mechanismu. Největší obratnosti dosahují řídicí algoritmy používající inverzní dynamiku. Ty používají výpočty polohy tzv. zero moment point, neboli bodu ve kterém se setrvačné momenty a momenty způsobené gravitací rovnají nule [14] [15]. Ke své činnosti potřebují přesné měření zrychlení těžiště mechanismu a měření sil mezi robotem a podlahou.

Za nejjednodušší řízení kráčejících mechanismů můžeme považovat přehrávání předem definované sekvence pohybů, pro chůzi v různých směrech. Takovéto řízení se používá například v animatronických systémech využívaných v kinematografii nebo v méně profesionálních podmínkách jako jsou hračky a stavebnice kráčejících robotů.

2 Požadavkový list

Při zadání diplomové práce byly určeny povinné a vedlejší úkoly, které popisují požadavky na výsledný systém.

Povinné úlohy:

- Čtyřnohý kráčející systém na principu bioorganismů,
- Senzorický subsystém pro detekci vnitřních stavů zařízení,
- Parametrický matematický model mechanismu,
- Antikolizní systém pro robot,
- Vizualizace robotu s využitím V-Rep.

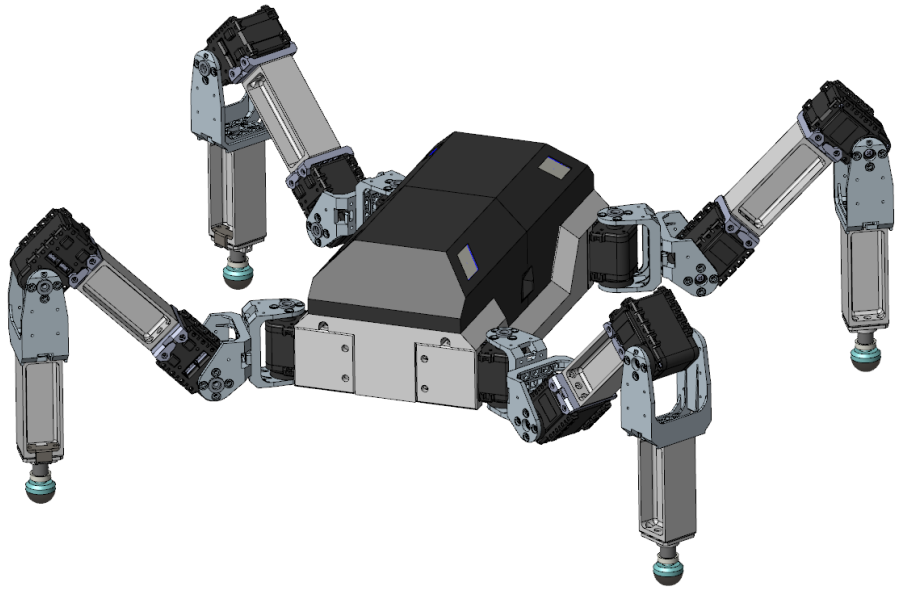
Vedlejší úlohy:

- Regulace pohonů v řídicí jednotce,
- algoritmy pro opětovné postavení se po pádu robota,
- Doplnění stavových LED,
- Ochranné kryty pro elektroniku a energetický zdroj,
- Režim napájení kabelem i pro napájení akumulátorem,
- komunikace je přes wire a wireless.

3 Mechanická část

Tato kapitola se zabývá popisem mechanické konstrukce a kinematiky robotu Marvin. Jeho jednotlivými částmi, konfiguracemi a kinematikou.

Jedná se o čtyřnohý robot s dvanácti stupni volnosti, schopný vykonávat složité pohyby. Každá noha má tři stupně volnosti. Jako pohony tohoto robotu jsou použity servomotory Dynamixel AX-12. Navrhované díly jsou přizpůsobeny pro výrobu z plastu na 3D tiskárně. Konstrukce dílů a model sestavy je vypracován v programu Creo Parametric. Robot je vybaven mikrokontrolerem a všechny výpočty potřebné pro pohyb jsou prováděny uvnitř robotu. Robot je osazen senzory monitorujícími stav robotu.



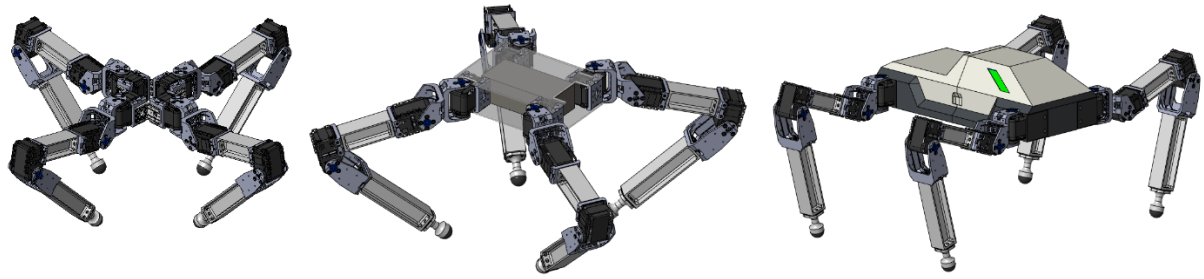
Obr. 5 Finální podoba konstrukce robotu Marvin

Tabulka 2 Základní vlastnosti robotu Marvin

Vlastnost	Hodnota	Jednotka
Celková hmotnost	1850	g
Rozměry těla	240x120x90	mm
Délka nohy (výchozí)	350	mm
Maximální rychlost kráčení	160	mm/s
Maximální rychlost klusání	400	mm/s
Výdrž provozu na baterii	60	min

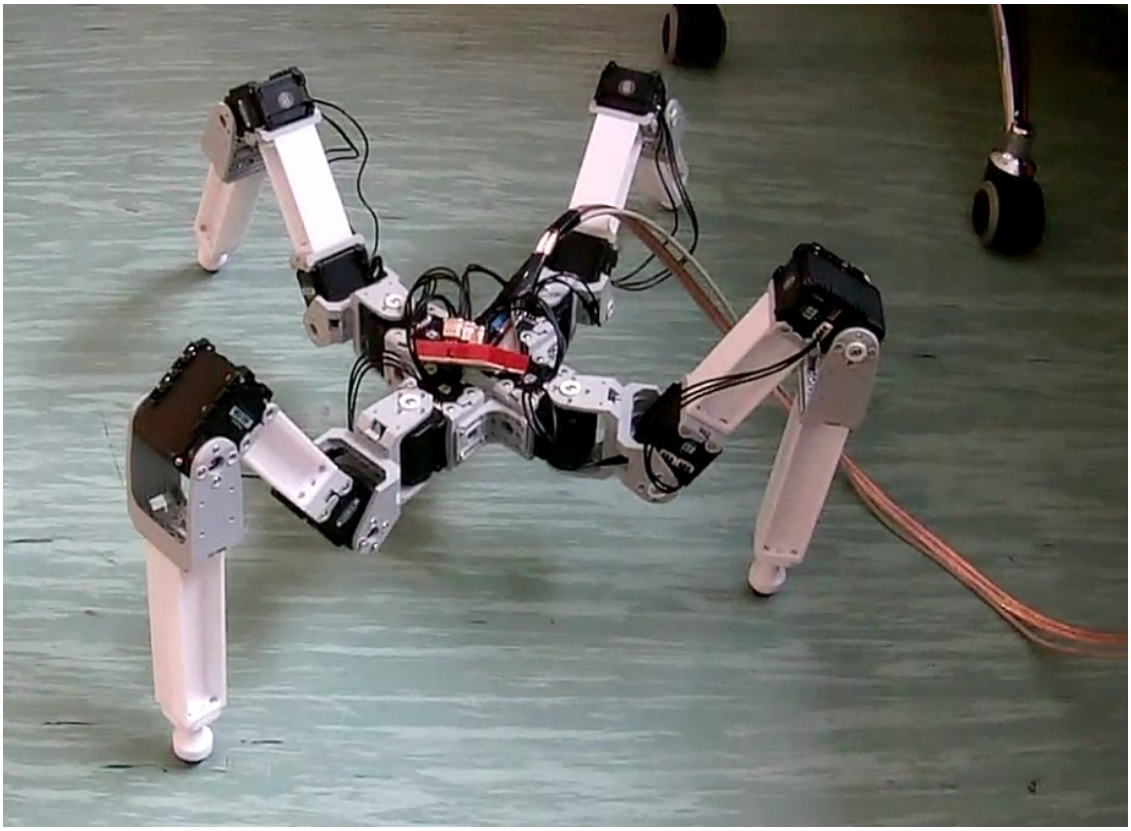
3.1 Iterace návrhu

Při konstrukci mechanismu jsem prošel několika iteracemi, ve kterých jsem postupně přidával funkce a vlastnosti robotu (Obr. 6).



Obr. 6 Bývalé iterace konstrukce robotu

První verze robotu byla složena pouze ze čtyř noh, propojených mezi sebou plastovými konzolami. Umístění dalších komponentů by mohlo být řešeno postupným přidáváním na střed robotu, nad první servomotory noh. Řídicí systém a napájení jsou v této verzi umístěny mimo robot.



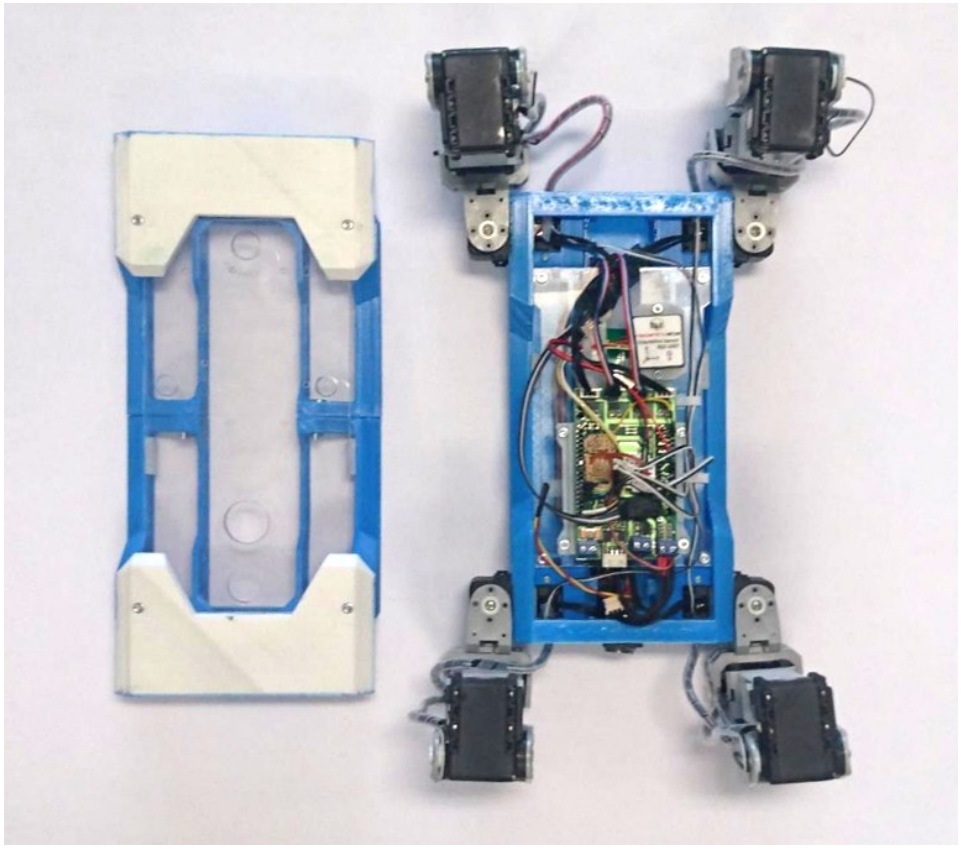
Obr. 7 Kráčení první verze robotu

V druhé verzi byly nohy posunuty dále od sebe, což zvýšilo celkovou stabilitu robotu a umožnilo umístění komponentů do prostoru mezi ukotvením noh. Tato verze tedy umožnila umístění řídicího mikrokontroleru dovnitř robotu a zároveň přesun výpočtů pohybu a

kinematiky z PC do robotu, což snížilo množství přenesených dat. Napájení bylo v této verzi stále řešené připojením kabelu z laboratorního zdroje.

Třetí verze je navrhována tak aby se do ní vešel akumulátor a veškerá elektronika. Je složena z dílů vyrobených na 3D tiskárně. Tyto díly jsou mezi sebou spojeny šroubovým spojem. Je zde možnost připojení kabelu k propojení s laboratorním zdrojem a počítačem. Nabízí možnost přestavění kinematiky na variantu hmyz nebo savec. Elektronika robotu je v této verzi krytována ze všech stran.

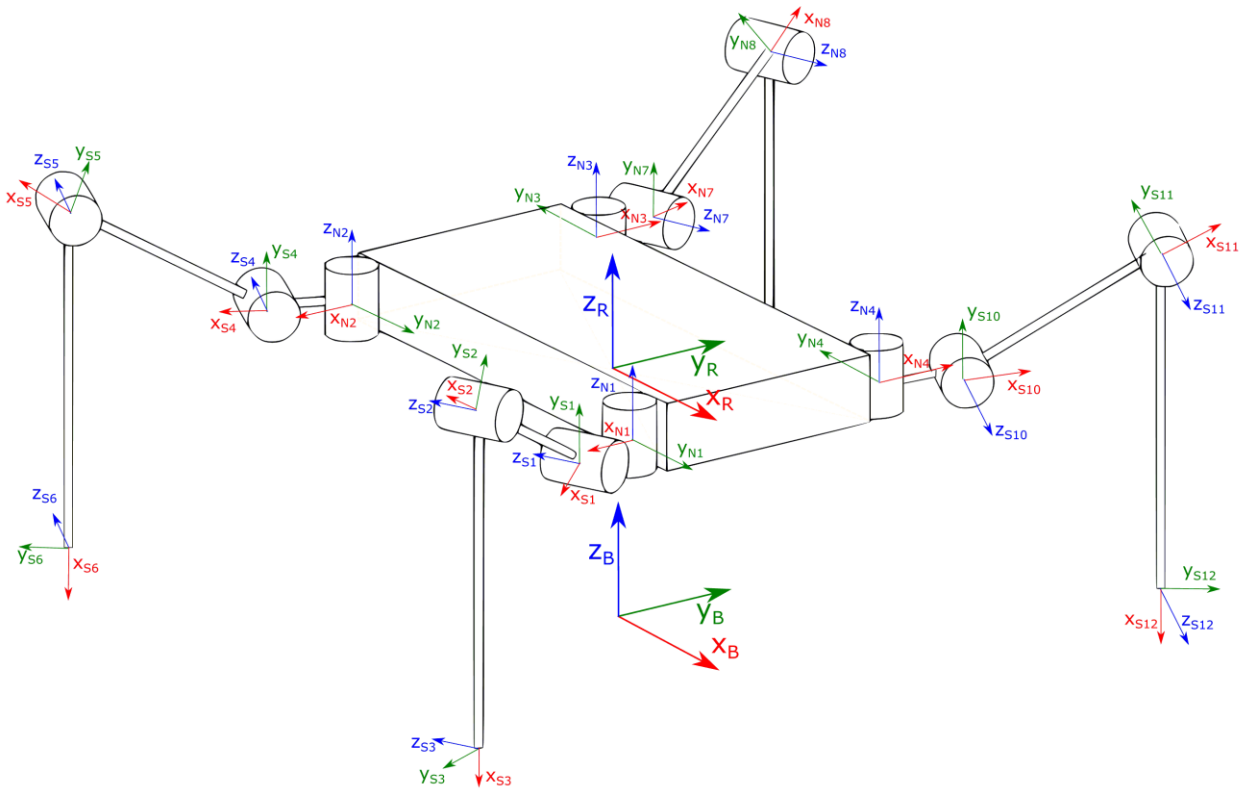
Čtvrtá a aktuální verze konstrukce těla je upravená třetí verze, která byla zmenšena a zjednodušena (Obr. 8). Obsahuje všechny komponenty a vlastnosti třetí verze a zároveň jsou v ní přidány senzory doteku a jednotka IMU. Z původních rozměrů třetí verze 300x140 mm došlo u této verze k zmenšení na 240x120 mm a tím k lepšímu poměru mezi tělem a nohami. Zjednodušením a spojením několika tištěných součástí v jednu, došlo k odlehčení konstrukce o 200 gramů.



Obr. 8 Porovnání velikosti těla třetí verze (vlevo) a čtvrté verze robotu (vpravo)

3.2 Kinematika

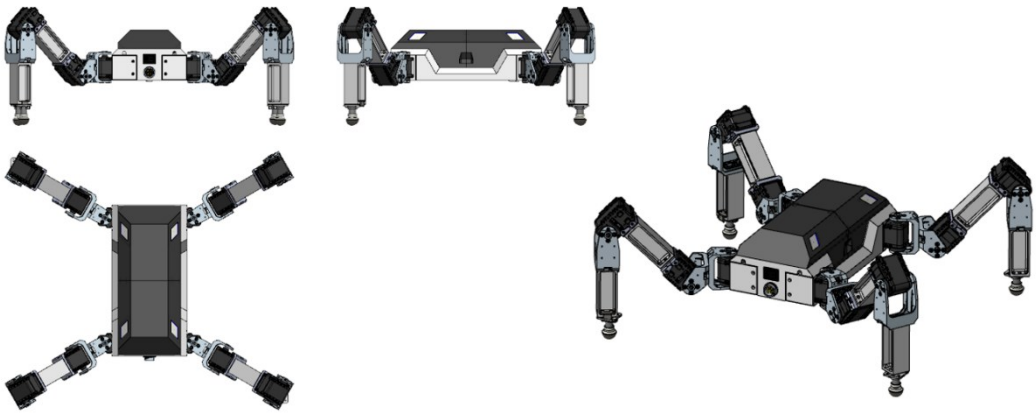
Robot má 12 stupňů volnosti a 13 vzájemně pohybujících se těles. Pro zjednodušení výpočtů bylo do těla robotu umístěno 5 souřadnicových systémů a jeden SS se nachází mimo robot. Pro výpočty kinematiky a další výpočty, jako je výpočet těžiště nebo detekce kolizí bylo v mechanismu zavedeno celkem 18 souřadnicových systémů (Obr. 9). Bázový souřadnicový systém S_B s osami [X_B, Y_B, Z_B] je základním referenčním rámcem pro výpočty a je umístěn na podlaze pod tělem robotu. Tělo má uprostřed své spodní stěny umístěn souřadnicový systém S_R s osami [X_R, Y_R, Z_R], tento SS je důležitý při výpočtu inverzní kinematiky, neboť tělo robotu může být libovolně posunuté a natočené oproti S_B. Dalšími souřadnicovými systémy jsou S_{N1}, S_{N2}, S_{N3} a S_{N4}, které se nacházejí v prvních osách rotace jednotlivých noh. Poloha a orientace těchto SS se mění v závislosti na konfiguraci mechanismu. Ostatní SS jsou definovány v pohyblivých segmentech nohy S_{S1} až S_{S12}. Tyto SS jsou voleny podle Denavit-Hartenbergova principu a tím zjednodušují přepočet bodů mezi jednotlivými systémy.



Obr. 9 Marvin - Souřadnicové systémy

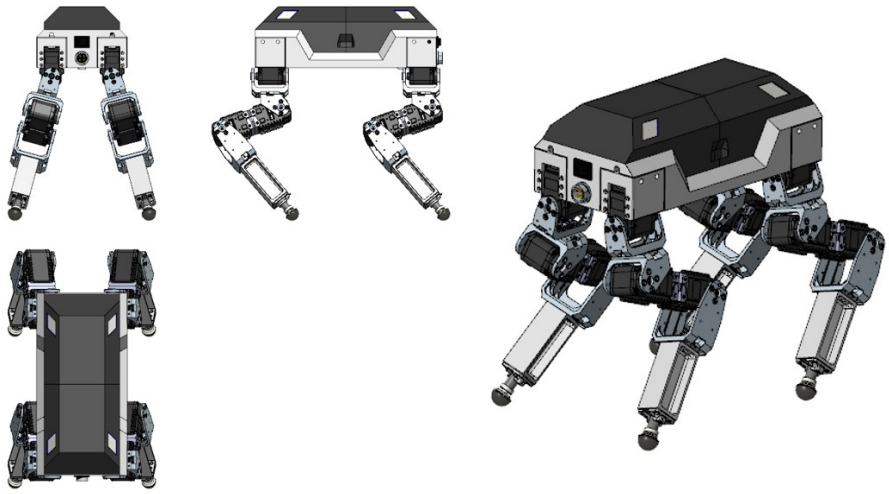
3.3 Konfigurace kinematické struktury

Konstrukce robotu je inspirována živými organismy. Robot je možné sestavit do dvou kinematických konfigurací, jedna inspirovaná rozložením končetin u hmyzu (Obr. 10) a druhá stavbou podobná savcům (Obr. 11) ..



Obr. 10 Kinematická struktura inspirovaná hmyzem

Kinematická struktura inspirovaná hmyzem má níže položené tělo, může se tedy dostat do nižších prostorů. Poloha těžiště blíže zemi pomáhá v udržení stability. Široké rozložení noh umožňuje tělu robotu náklon těla až 40°kolem jakékoliv osy.

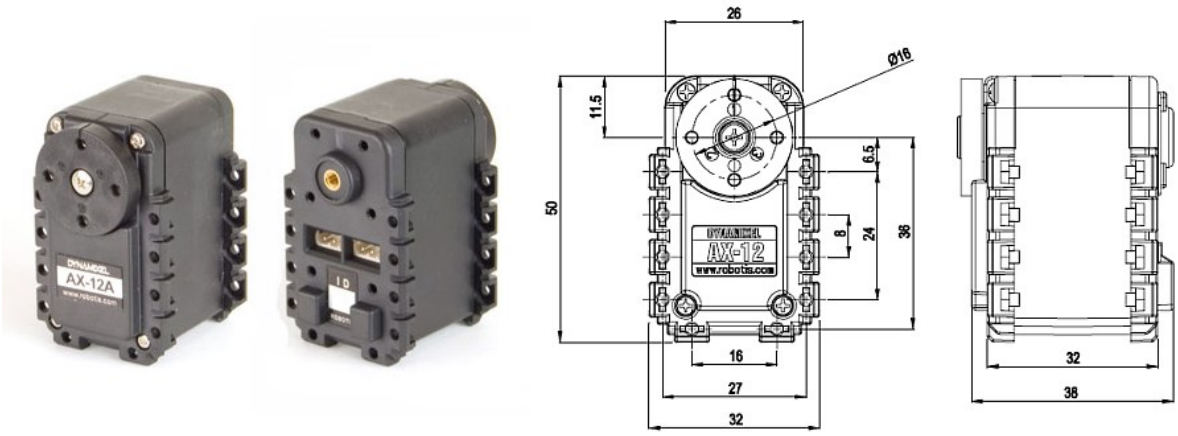


Obr. 11 Kinematická struktura inspirovaná savci

Kinematická struktura inspirovaná savci má nohy umístěné pod tělem. Jejich orientace je podobná zadním nohám koně. Při pohybu v této konfiguraci se nohy pohybují o menší vzdálenosti. Statické i dynamické momenty působící na servomotory jsou menší. Momenty vyvolané zvedáním noh pod robotem mají menší tendenci překlápět robot do strany než při zvedání noh vyložených bokem. Vyšší položení těla robotu je náchylnější na nerovnosti povrchu, po kterém robot kráčí, zároveň ale umožňuje zvednutí nohy a překročení vyšší překážky než v konfiguraci inspirované hmyzem.

3.4 Pohony

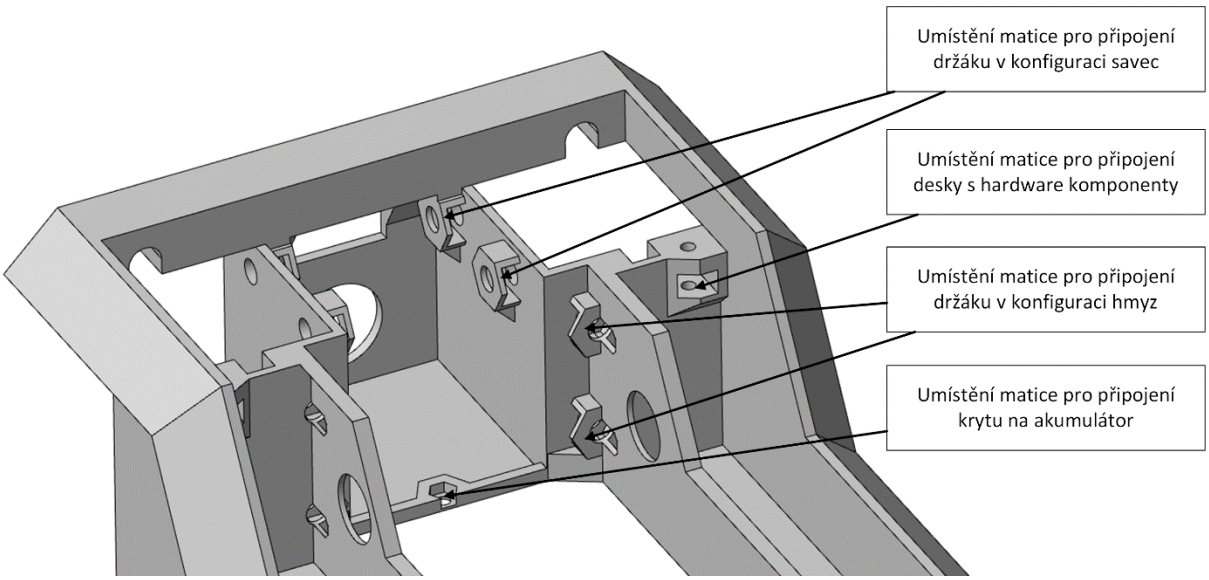
Základem konstrukce robotu jsou servomotory Dynamixel AX-12 [16]. Tyto servomotory jsou napájeny napětím 12V. Jsou schopny poskytnout krouticí moment až 1,5 Nm při maximální spotřebě 900 mA. Servomotory Dynamixel jsou řízeny přes sériovou sběrnici, což je výhoda oproti klasickým servomotorům řízeným pulzní šířkovou modulací, neboť pro jejich řízení stačí pouze jeden vodič. Přes přístup k vnitřním registrům poskytují údaje o aktuální poloze, působeném krouticím momentu nebo teplotě motoru. Na Obr. 12 jsou zobrazeny rozměry motoru, ze kterých vychází návrh celého mechanismu.



Obr. 12 Servomotor Dynamixel AX-12 a jeho rozměry

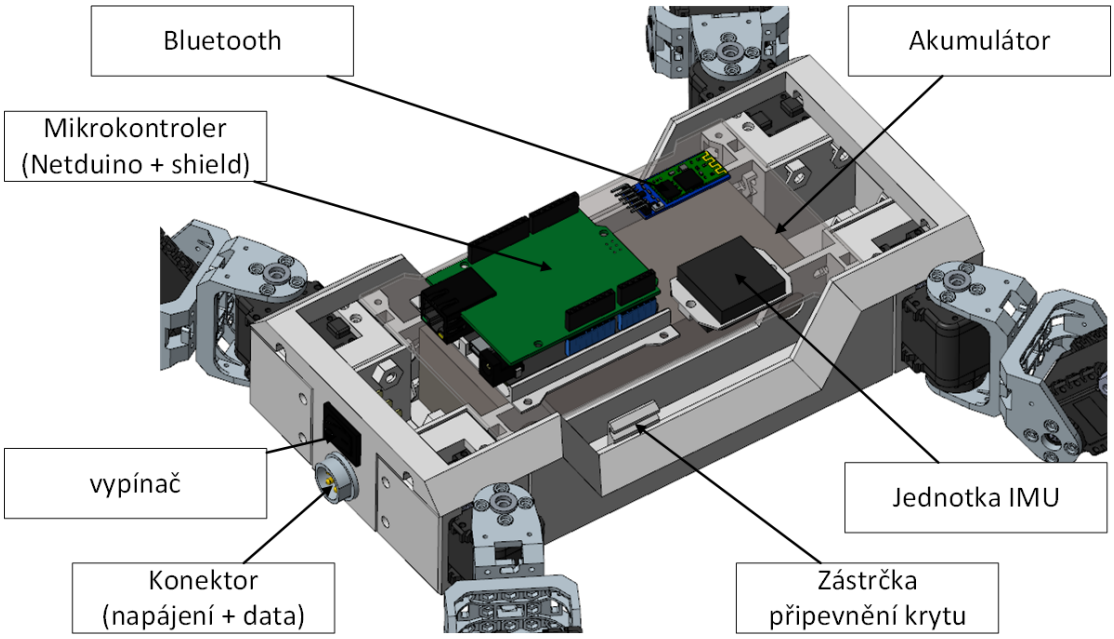
3.5 Tělo robotu

Tělo robotu tvoří konstrukce vytištěná z materiálu PLA na kterou jsou připojeny ostatní komponenty. Hlavními požadavky na tento díl byla možnost připevnění noh ve dvou orientacích, dostatek prostoru pro uložení baterie, přístupné místo pro konektor kabelu a vypínač a jednoduše snímatelný kryt. Pro odlehčení je tělo vytištěno jako dutá skořápka s dvouvrstvou stěnou. Toto odlehčení nemá výrazné zhoršení pevnosti při porovnání s výtisky s vnitřní strukturou a výhoda snížené hmotnosti převyšuje sníženou tuhost. Spojení tištěných dílů je řešeno šroubovým spojem s vloženými nebo vlepenými maticemi. Pouhé závity vyřezané v tištěném plastu mají velmi omezenou životnost. Tělo má proto v sobě speciálně tvarované části, do kterých se matice vloží (Obr. 13).



Obr. 13 Umístění vložených matic v těle robotu

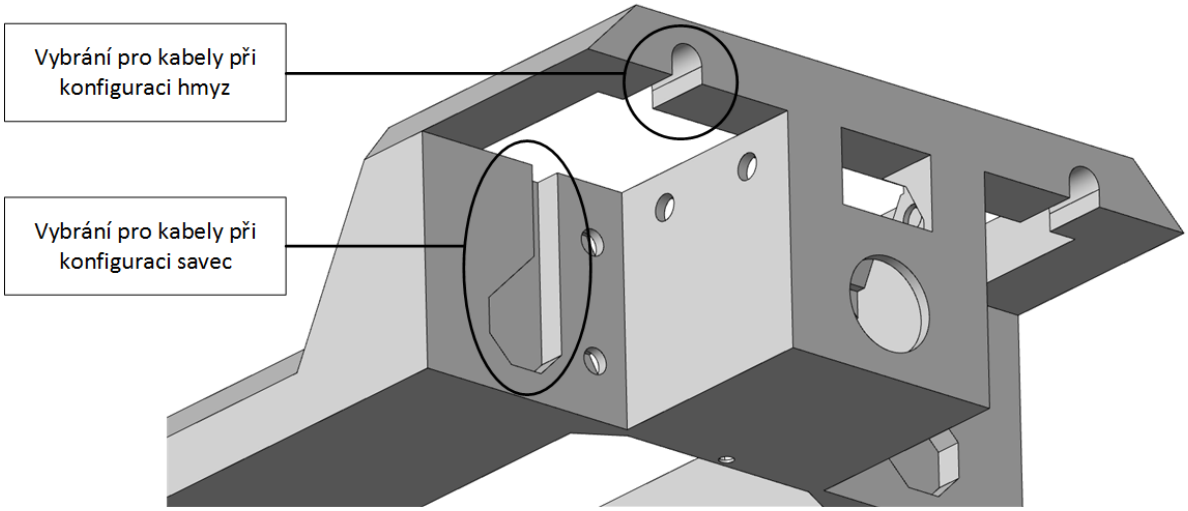
Většina hardwarových komponentů je umístěna pod krytem (Obr. 14). Pro jednoduchost a možnost dále upravovat rozmístění komponentů, jsou Netduino, jednotka IMU, modul Bluetooth a kabelové držáky umístěny na plexisklové desce, která je připevněna k tělu robotu. Tělo robotu je navrženo tak, aby poskytovalo dostatek prostoru pro vedení kabelů uvnitř konstrukce.



Obr. 14 Pohled pod kryt robotu

Dvířka pro akumulátor jsou umístěny na spodní straně těla robotu. Přístup ze spodní strany umožňuje rychlou výměnu akumulátoru, bez toho, aniž by mohlo dojít k zásahu do zbytku kabeláže robotu.

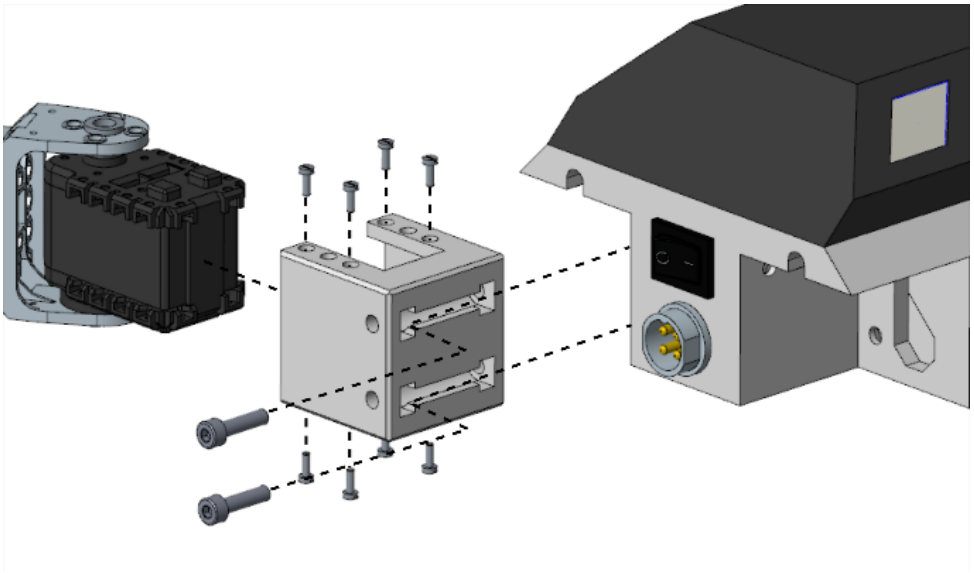
Svazek kabelů vedoucí do nohy může vést různými místy, podle aktuální konfigurace mechanismu. Jedná se o svazek obsahující připojení servomotorů a senzorů dotyku. V těle robotu byly proto vytvořeny vybrání pro vedení těchto kabelů, tak aby nepřekáželi při připojení držáků servomotorů ani v jedné konfiguraci ().



Obr. 15 Přispůsobení těla robotu pro vedení kabelů do noh

3.6 **Držáky servomotorů**

Servomotory jsou v těle připevněny komocí tištěných držáků. Ty jsou svým tvarem krychle o rozměru 40x40x40 přizpůsobeny tak, aby se daly do těla připevnit v několika konfiguracích. Servomotor je k držáku přichycen osmi šrouby M2x6. Umístění šroubů M4x12 dovoluje oboustranné použití. Drážka, do kterých se vkládají šrouby M4, je tvarována tak aby při montáži a změně orientace šroub nevypadával.



Obr. 16 Držák servomotoru umožňující přestavění

3.7 **Kryt**

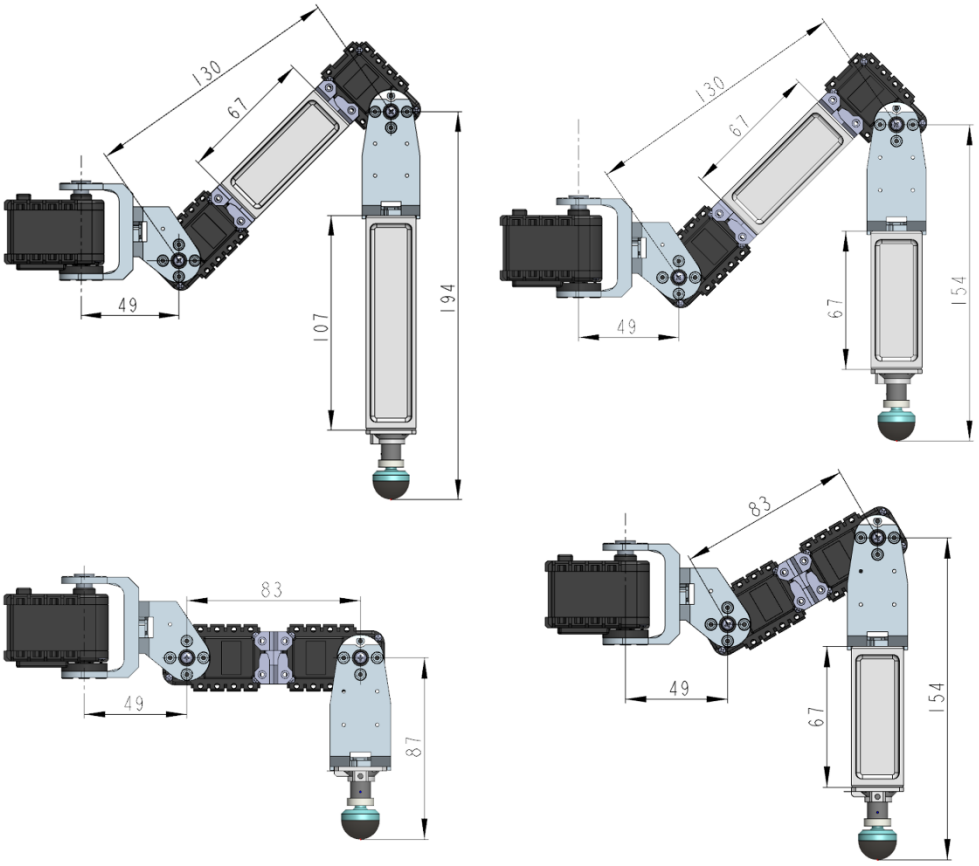
Kryt robotu poskytuje ochranu proti zásahu do kabeláže a hardwarových komponent v robotu. Jsou v něm vlepeny průsvitné součásti, které jsou určené pro uložení stavových LED světel. Musel být kvůli svému tvaru vytištěn jako dvě poloviny a poté sestaven do jednoho celku. Kryt je k tělu robotu připevněn tvarovým stykem. Je dost pružný na to, aby se mohl opakovaně deformovat a jednoduše nasazovat na tělo robotu.

Obr. 17 Sestavený kryt robotu

Obr. 18 Kryt – dolní pohled

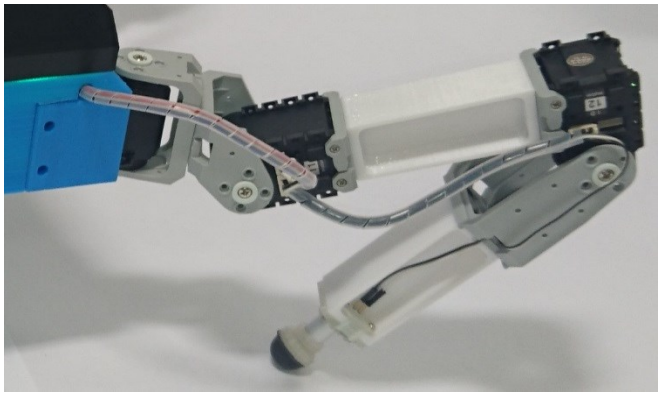
3.8 **Noha**

Noha robotu je složena ze tří segmentů, spojených mezi sebou servomotory Dynamixel. Obsahuje plastové konzoly servomotorů ze stavebnice Bioloid a tištěné sloupky. Výměnou těchto tištěných sloupků lze upravovat délku nohy a tím měnit kinematiku mechanismu. Byly vytisknuty dvě velikosti sloupků s délkami 67 mm a 107 mm. Na Obr. 19 jsou zobrazen některé kombinace sestavení nohy s použitím těchto sloupků, nebo bez.



Obr. 19 Různé kombinace sestavení nohy

Kabely vedoucí k servomotorům v noze a k senzoru na konci nohy jsou umístěny v kabelových vodičích (Obr. 20).



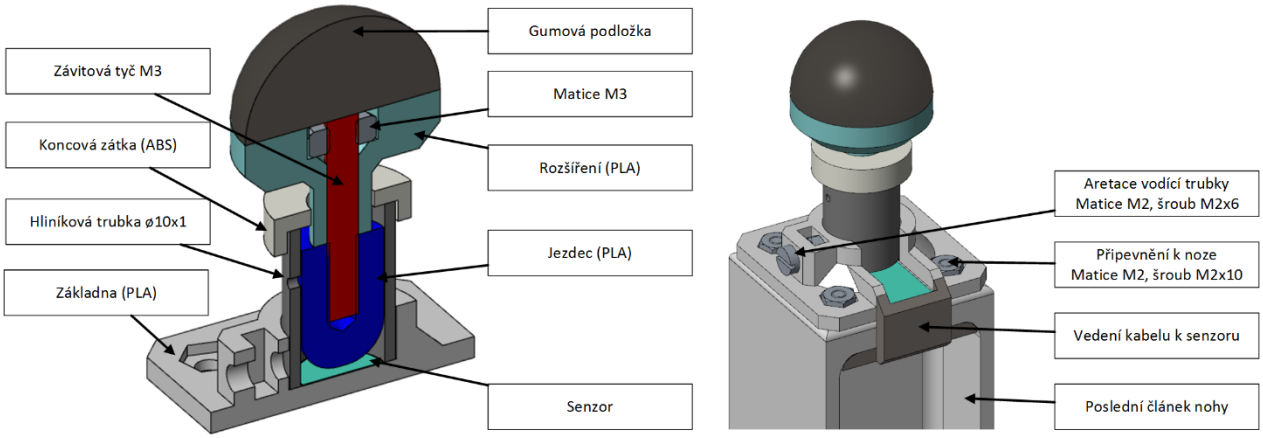
Obr. 20 Noha - vedení kabelů

3.8.1 **Senzory doteku**

Pro detekování kontaktu nohy s podlahou jsou použity senzory FSR 400 (Force-Sensing Resistor) [17]. Tyto senzory mění svůj odpor na základě působené síly na jejich plochu a mají použitelný rozsah od 0N do 15N. Tento senzor se vyrábí v několika velikostech, pro použití na konci nohy byla zvolena nejmenší varianta s průměrem 5mm (Obr. 21).

Obr. 21 Použitý senzor FSR

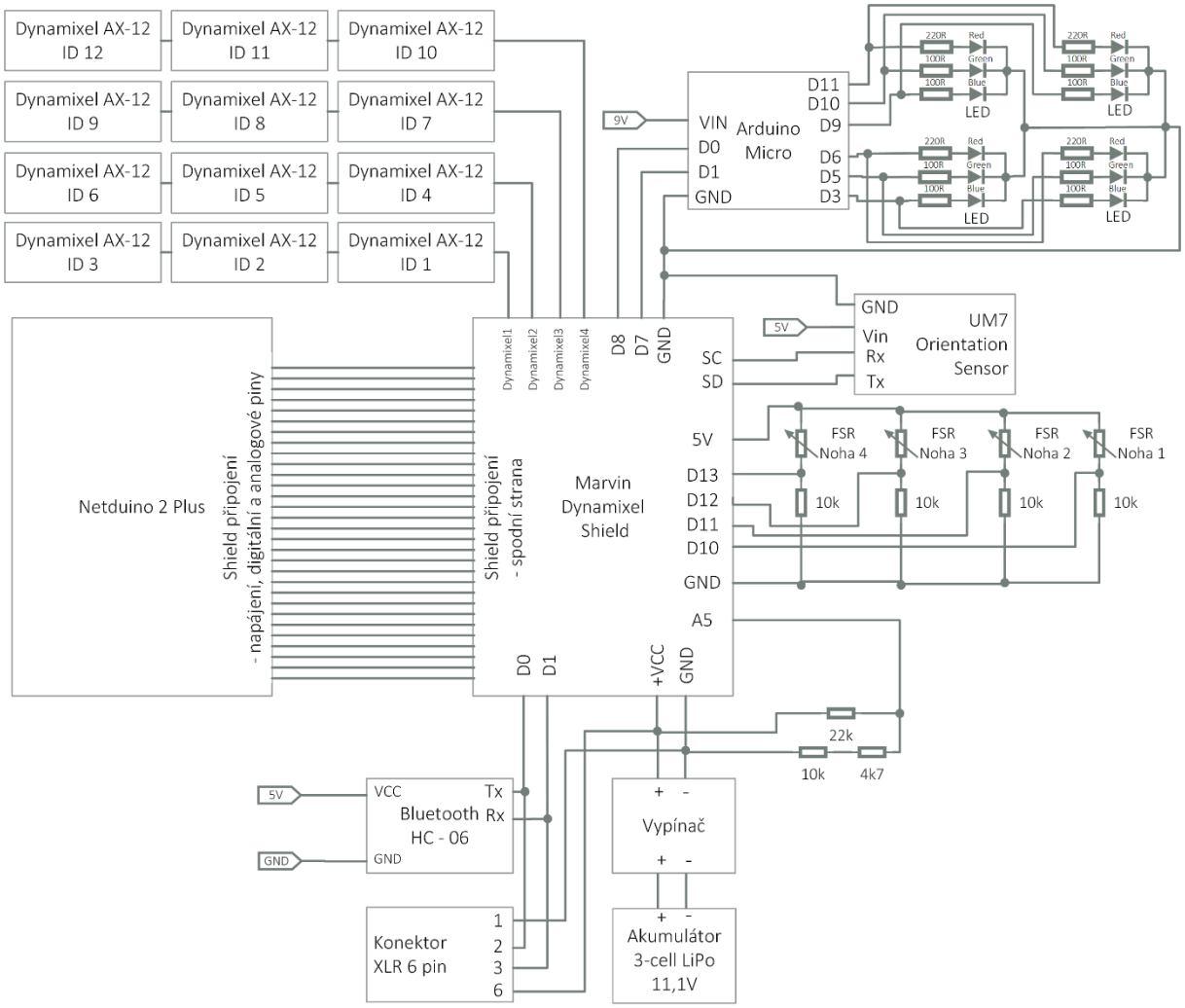
Pro tento senzor bylo vyrobeno uložení s posuvnými prvky, které na senzor doléhají a při kontaktu nohy s podlahou jsou přitisknuty k senzoru (Obr. 22). Pohyblivý jezdec je složen z několika dílů tak, aby bylo sestavu možné smontovat. Jezdec je uložen v hliníkové trubce, která díky svému hladkému povrchu nepůsobí na jezdec velkým třením a nezamezuje mu pohyb.



Obr. 22 Uložení senzoru FSR v chodidlu nohy

4 Hardwarová část

Tato kapitola popisuje jednotlivé hardwarové komponenty v robotu. Použitý mikrokontroler Netduino, jeho rozšíření a připojené senzory.



Obr. 23 Schéma zapojení

4.1 Netduino

Mozkem robotu je deska Netduino Plus 2. Netduino je open-source elektronická platforma pro prototypování. Obsahuje 16 programovatelných digitálních pinů, 6 analogových vstupu, 4 UART sběrnice, Ethernetový RJ45 port a slot na MicroSD karty. Deska je navržena jako pinově kompatibilní s většinou shieldů pro Arduino. [18] Netduino lze napájet napětím 7,5V až 9V a obsahuje regulátory napětí s výstupy 5V a 3,3V [19].

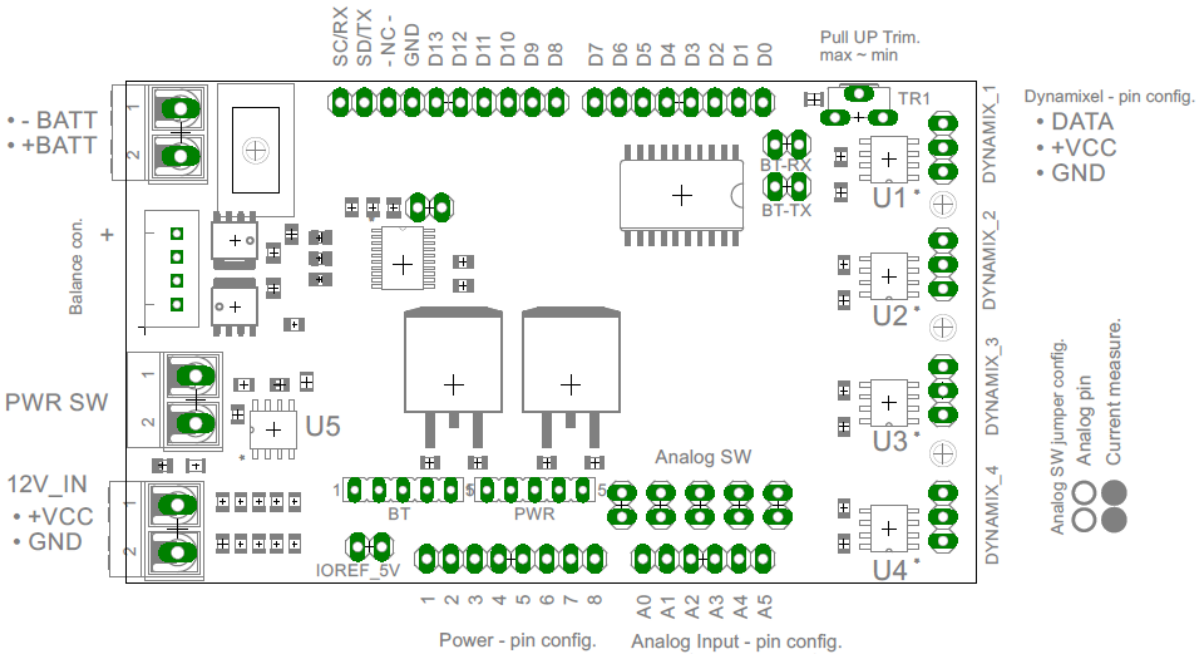
V tomto zapojení (Obr. 23) je Netduino napájeno přes připojený shield, který poskytuje regulovaných 9V. Veškeré digitální a analogové vstupy a výstupy jsou rovněž připojeny přes na míru vyrobený shield.

4.2 Shield

Netduino je rozšířeno připojeným shieldem, který byl podle mnou zadaných specifikací vyroben na katedře Robotiky. Hlavním účelem použití shieldu bylo přidání podpory ovládání servomotorů Dynamixel. Shield je nasazen na desce Netduino a veškeré další připojené komponenty se připojují přes něj.

Shield obsahuje integrovaný buffer SN74LS241DW, který z dvouvodičové sériové linky dělá jednovodičovou, kterou používají servomotory Dynamixel. Toto připojení využívá druhou sériovou linku na pinech D2 a D3, zároveň využívá pin D6 pro přepínání mezi čtením a odesíláním dat na linku.

Deska umožňuje měření celkového odebíraného proudu a proudu na jednotlivých nohách. K tomu slouží čtveřice Hallových sond ACS712, měřicí odebírané proudy jednotlivých noh, a jednu sondu ACS711, měřicí celkový proud. Hallovy sondy mají výstup v podobě analogového signálu a jsou na shieldu připojeny k analogovým pinům A0 až A4. Toto připojení je vedeno přes vyjímatelný jumper, díky kterému se dají Hallovy sondy odpojit v případě potřeby využití analogových pinů pro jiné účely.

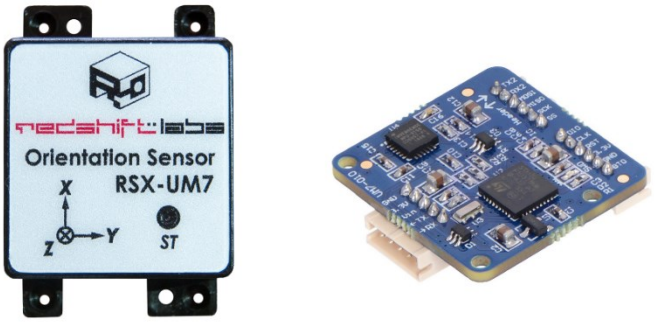


Obr. 24 Marvin Dynamixel Shield

Shield je osazen dvěma regulátory napětí, které vytváří napětí 9V a 5V. Napětí 9V je přivedeno na pin VIN a tím napájí Netduino. Napětí 5V lze použít pro napájení senzorů a je přivedeno na pin IOREF na Netduinu. Servomotory jsou napájeny přímo z akumulátoru nebo připojeného kabelu, bez regulace napětí.

4.3 IMU

K měření orientace robotu v prostoru slouží jednotka IMU (Inertial Measurement Unit). Použitý typ UM7 od firmy Redshift labs [20], filtruje a kombinuje měření z akcelerometru a gyroskopu. Výsledkem je orientace robotu v podobě Eulerových úhlů nebo kvaternionu. Senzor poskytuje široké možnosti nastavení parametrů měření a odesílání měřených dat. Je napájen napětím 5V se spotřebou 50mA, komunikuje přes 3,3V UART nebo sběrnici SPI s nastavitelnou frekvencí odesílaní dat od 1Hz do 255Hz.



Obr. 25 UM7 Orientation Sensor s krytem a bez krytu

4.4 Stavové světla

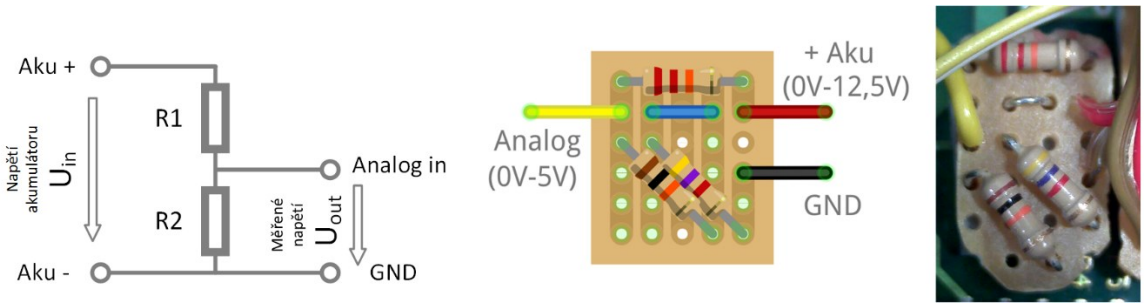
Pro znázornění vnitřních dějů v programu robotu jsou v krytu robotu umístěny stavové RGB LED. Ke každé diodě je připojen sériově rezistor, tak aby omezoval odebíraný proud na přibližně 15 mA. RGB LED jsou připojeny paralelně v párech pro přední a zadní světla.

Připojení LED světel umístěných v krytu robotu je pro nedostatek pinů na Netduinu řešeno přes mikrokontroler Arduino micro. Toto připojení využívá pouze dva piny na Netduinu a umožňuje použít šest digitálních pinů s pulzní šířkovou modulací na Arduinu. Arduino zároveň dovoluje odebírat větší proud z jednotlivých výstupních pinů, až 40mA oproti 25mA na Netduinu. Piny D3, D5 a D6 ovládají přední světla a piny D10, D11 a D12 ovládají zadní světla.

Obr. 26 Rozsvícené stavové LED

4.5 Dělič napětí

Analogové vstupy desky Netduino měří připojené napětí od 0V do referenčního napětí, které v tomto případě 5V. Proto je potřeba při měření napětí na baterii nejprve napětí snížit. K tomuto účelu byl sestaven dělič napětí. Rezistory R1 a R2 jsou voleny tak aby se co nejvíce přiblížily výstupnímu napětí 5V a vstupnímu napětí přes 12V. Podle vzorce (2) jsou vybrány odpory R1 = 22kΩ a R2 = 14,7kΩ, složený ze dvou sériově zapojených rezistorů o hodnotách 10kΩ a 4,7kΩ.



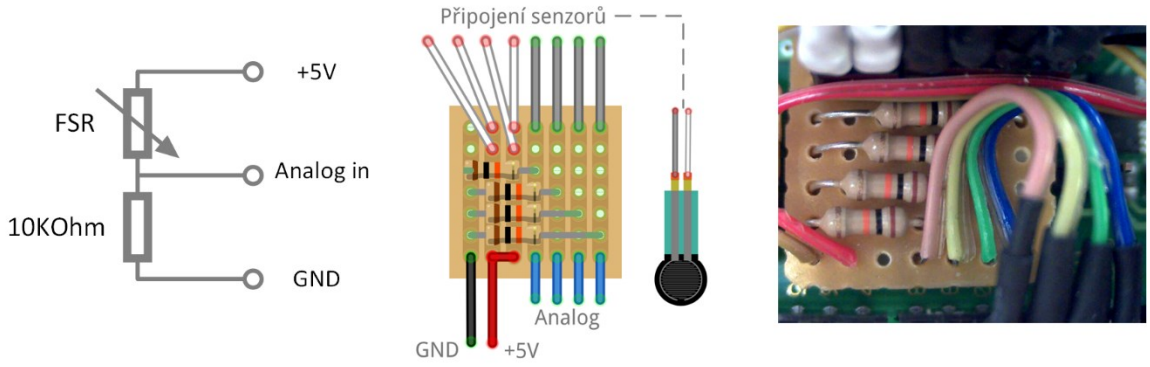
Obr. 27 Dělič napětí. Schéma (vlevo) a vyrobená deska (vpravo)

$$U_{out} = \frac{U_{in} \cdot (R_1 + R_2)}{R_2} \tag{1}$$

$$U_{out} = \frac{5 \cdot (22 + 14{,}7)}{14{,}7} = 12{,}483\ V \tag{2}$$

4.6 Senzory doteku

Senzory dotyku [17] mění svůj odpor v závislosti na působení síly na jejich plochu. Použité senzory mají použitelný rozsah od 0N do 15N. Připojení těchto senzorů vyžaduje použití pull-down rezistoru, s doporučenou hodnotou 10Kohm. Při zatížení klesá odpor senzoru a tím se zvyšuje proud protékající přes senzor a rezistor. Tím se zvýší napětí na rezistoru, měřením tohoto napětí tedy zjistíme zatížení senzoru. Protože všechny analogové vstupy na Netduinu jsou zaplněny měřením proudu a napětí, jsou pro měření dotyku použity namísto analogových vstupů, vstupy digitální. Tím pádem můžeme zjistit pouze binární informaci, je-li noha v kontaktu s podlahou nebo ne. Aby digitální vstup vyhodnotil vstupní napětí jako logickou 1, je třeba aby překročilo hodnotu 2V, což pro použité senzory a zapojení představuje sílu cca 3N. Pro zapojení senzorů byla vyrobena deska s pull-down rezistory.



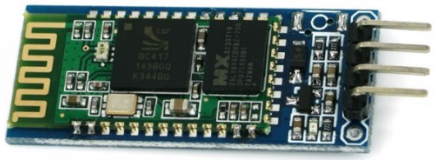
Obr. 28 Schéma zapojení senzoru dotyku (vlevo) a deska pro čtyři senzory (vpravo)

4.7 Připojení

Robot může s operátorem komunikovat dvěma způsoby. Bezdrátově nebo pomocí kabelu.

4.7.1 Bezdrátové

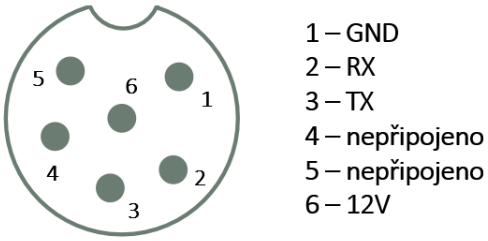
K bezdrátovému připojení je použit bluetooth modul HC-06. Modul je napájen napětím 5V se spotřebou kolem 8mA. Přenosová rychlost modulu je nastavitelná od 9600bps do 115200bps. Po spárování počítače s tímto modulem se v PC nainstaluje virtuální sériový port, přes který poté probíhá komunikace s robotem.



Obr. 29 modul Bluetooth HC-06

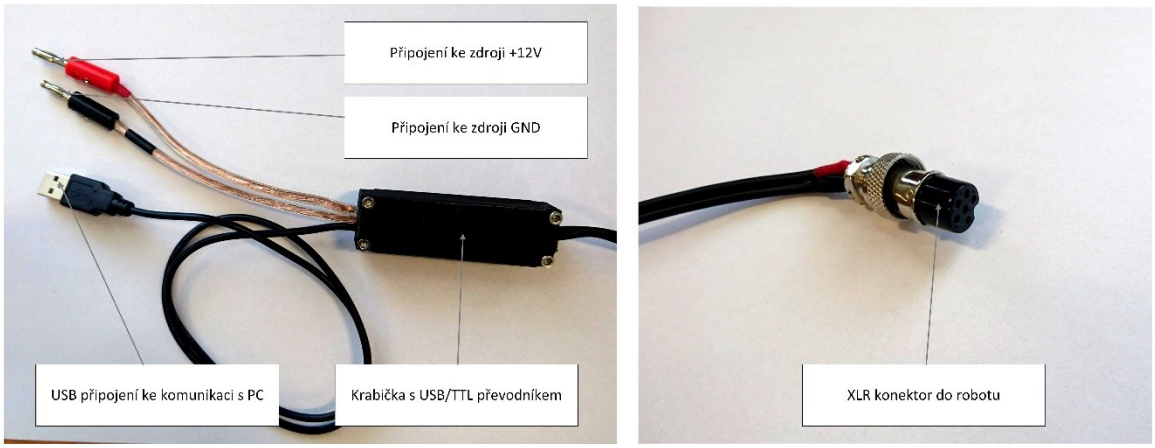
4.7.2 Kabel

Připojení kabelem může zároveň poskytovat napájení robotu a komunikaci s PC. Pro připojení k robotu je použit 6 pinový XLR konektor. V XLR konektoru jsou využity čtyři piny jako GND, +12V, RX a TX (Obr. 30).



Obr. 30 Využití pinů v konektoru robotu

Kabel má v sobě převodník USB/TTL, který je umístěn v krabičce na kabelu. Tento převodník má čtyři výstupní piny, RX, TX, +5V a GND. Napájení 5V z převodníku není použito a není tedy vedeno kabelem do robotu. Z krabičky vedou dva kabely pro připojení k laboratornímu zdroji. Červený pro +12V a černý k připojení k zemi. Černý kabel je uvnitř krabičky propojen s GND datového převodníku a do robotu pokračuje pouze jedno GND připojení.



Obr. 31 Připojovací kabel

4.8 **Akumulátor**

Tělo robotu je přizpůsobeno tak, aby se do něj vešel akumulátor o kapacitě 5200mAh [21]. Jedná se o tříčlánkový LiPo akumulátor s napětím 11,1V. Tento akumulátor je schopný napájet robot déle než hodinu. Hmotnost akumulátoru je 363g. Robot byl vyzkoušen i s menším akumulátorem o kapacitě 3000mAh, vážícím 269g. S menším akumulátorem měl robot stále použitelnou výdrž přes půl hodiny a výhodou byla větší stabilita při dynamické chůzi a více prostoru pro kabeláž uvnitř robotu.

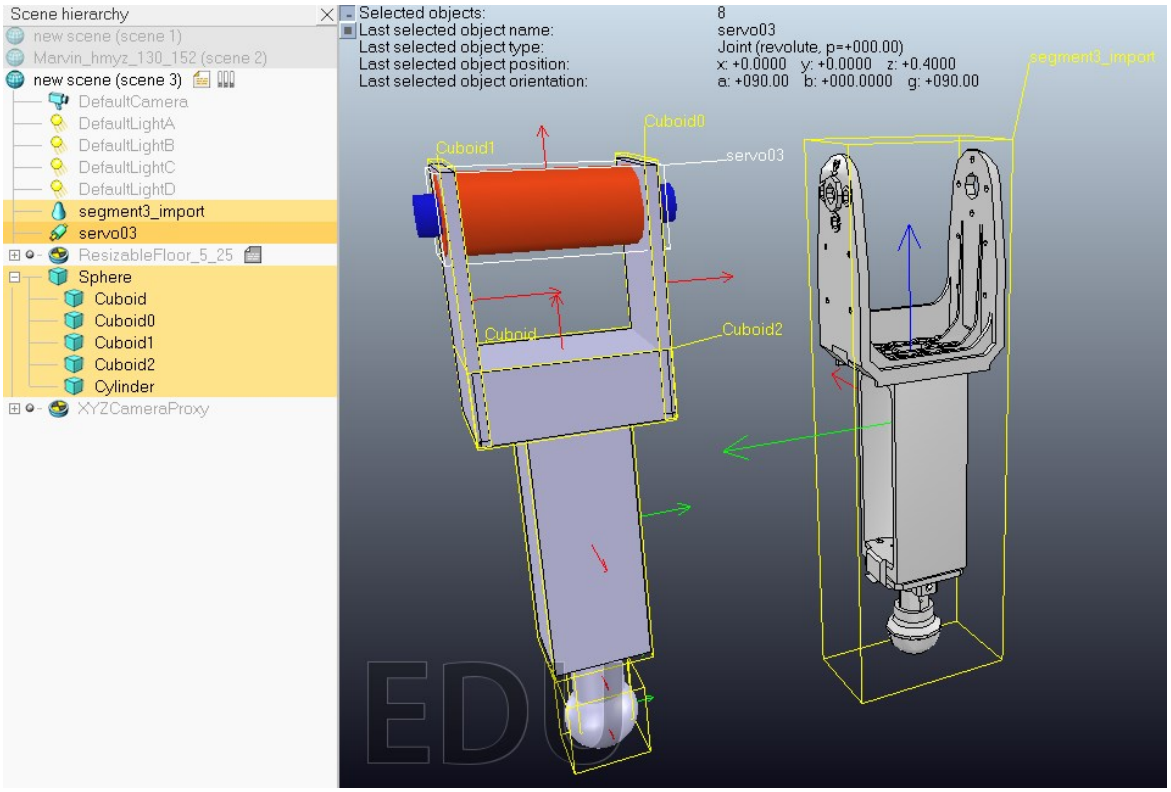
Obr. 32 Akumulátor 5200mAh

5 Simulace pohybu robotu

Vyzkoušení kráčejících a pomocných algoritmů může být v reálu zdlouhavé a hrozí riziko poškození fyzického robotu. Proto je výhodou vyzkoušet pohyb robotu nejprve v simulačním programu. Tato kapitola popisuje tvorbu simulačního modelu a využití simulací při návrhu algoritmů.

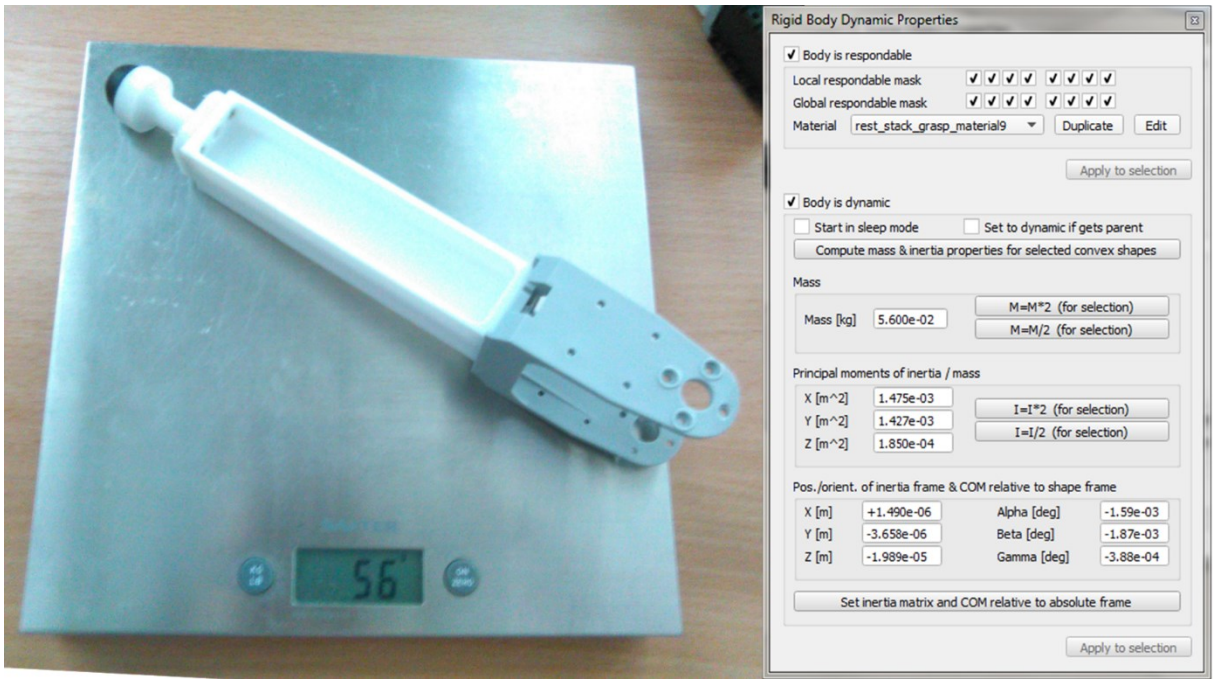
Pro simulaci mechanismu jsem použil program V-Rep od Coppelia Robotics [22]. V-Rep je multi-platformová aplikace, která umožňuje dynamické simulace objektů, vytvořených v integrovaném prostředí nebo importovaných z externí aplikace. Každý objekt v simulaci může být ovládán jednotlivě pomocí vloženého skriptu, pluginu, ROS uzlu nebo vzdáleného API klienta. To dělá z V-Rep všestranné řešení pro aplikace více robotů nebo interakci robotu v prostředí s dalšími předměty. Ovladače můžou být napsány v C/C++, Python, Java, Lua, Matlab nebo Octave.

V-Rep umožňuje import 3D modelů ve formátu .stl. Tyto modely jsou často tvarově složité a jejich simulace je náročná se špatnými výsledky. Proto je doporučeno používat importované modely jenom jako vizuální reprezentaci pohyblivého modelu při simulaci nebo jako statické objekty u kterých není třeba počítat dynamické účinky. Pro dynamickou simulaci importovaného modelu je třeba nahradit model skupinou základních tvarů jako jsou kvádry, válce a koule.



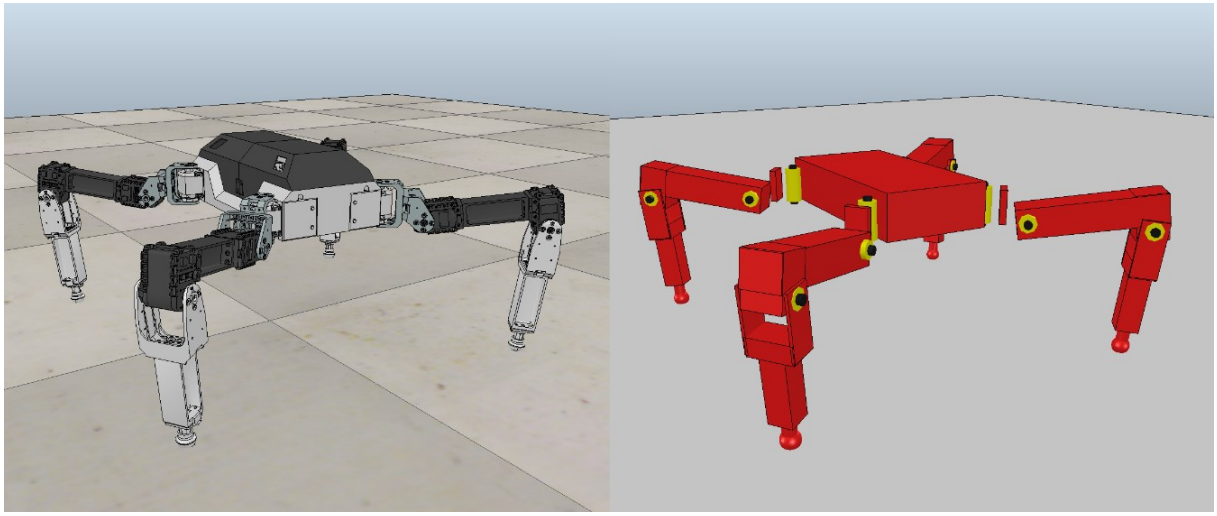
Obr. 33 Nahrazení importovaného modelu zkladními tvary

Po sestavení základních tvarů do podoby importované součásti se tyto tvary seskupí a nastaví se jim hmotnost součásti (Obr. 34). Aby se simulace co nejvíce přibližovala skutečnosti, lze nastavit i momenty setrvačnosti nebo součinitel tření na povrchu součásti.

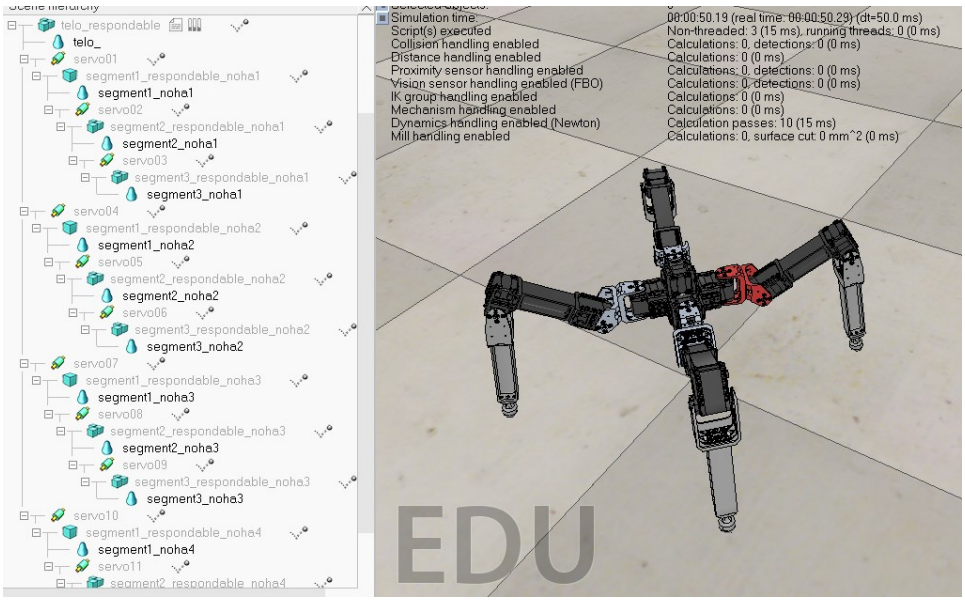


Obr. 34 Nastavení dynamických vlastností modelu

Vytvořené díly mechanismu se následně poskládají do podoby celého mechanismu (Obr. 36). Jednotlivé díly jsou v modelu hierarchicky spojovány pomocí rotačních kloubů, jejichž úhel bude při simulaci pohybu nastavován. Ve výsledku je model složen z importovaných objektů, sloužících k věrné vizuální reprezentaci robotu, a z objektů pro dynamické výpočty z jednoduchých tvarů. (Obr. 35).

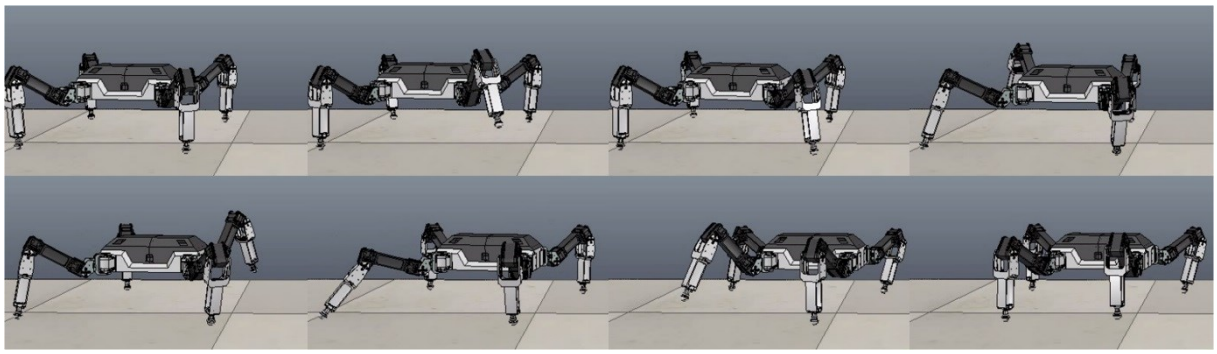


Obr. 35 Simulační model robotu (vlevo: vizuální, vpravo: dynamický)

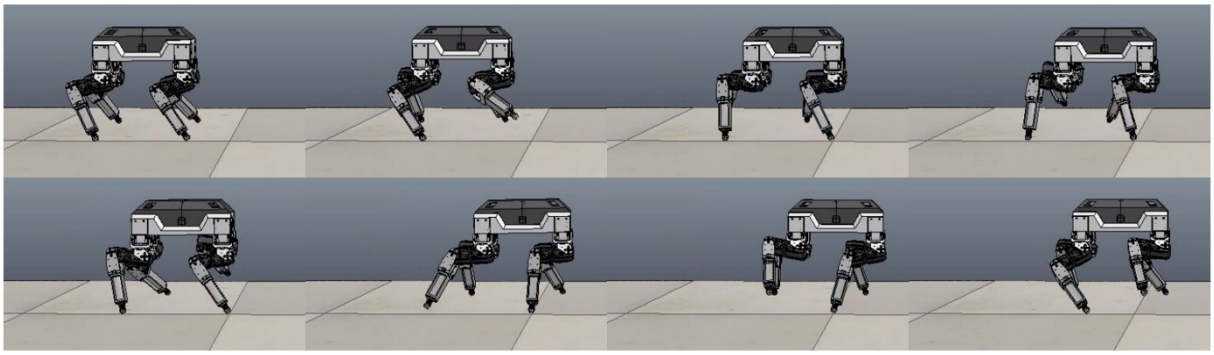


Obr. 36 První simulační model robotu

Velkou výhodou použití simulací při návrhu algoritmu kráčení a inverzní kinematiky byla možnost tyto algoritmy vyzkoušet na různých kinematických konfiguracích bez nutnosti přestavování samotného robotu. Použití více simulačních modelů pro různé kinematické konfigurace zároveň odhalilo chyby způsobené špatným poskládáním modelu, nebo chyby ve výpočtech kinematiky, které se projevovaly jenom v některých konfiguracích. Na Obr. 37 a Obr. 38 lze vidět, jak stejný algoritmus kráčejícího pohybu ovládá dva různé simulační modely a dokazuje tak jeho širší využití.

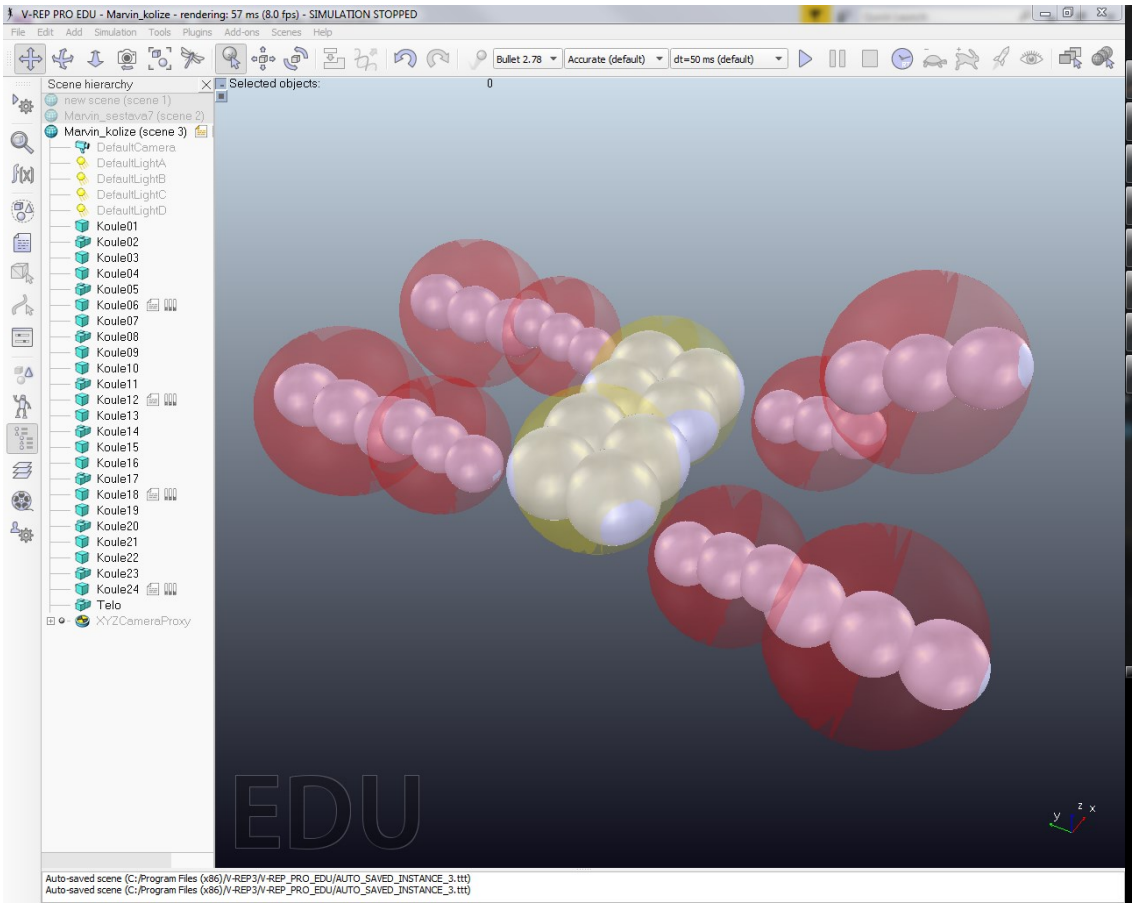


Obr. 37 Kráčení v konfiguraci hmyz



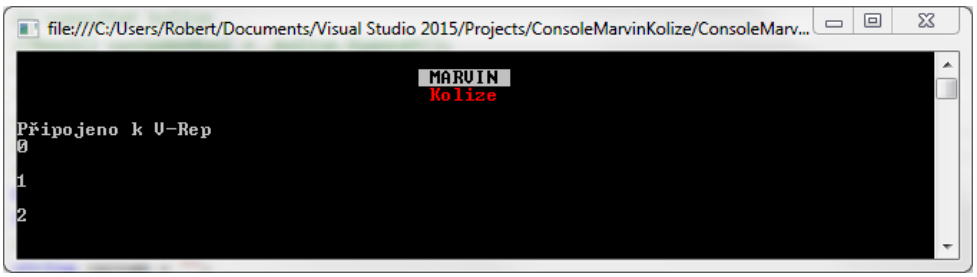
Obr. 38 Kráčení v konfiguraci savec

Simulaci jsem využil i při návrhu algoritmu pro kontrolu kolizí, kde jsem model robotu nahradil kulovými objemy. Tato kontrola počítá polohy jednotlivých objemů a rozhoduje, jestli se vzájemně překrývají nebo ne. K posouzení, zda kontrola opravdu funguje tak jak má, je třeba robot uvést do kolize a sledovat, jak se výpočet zachová. Ručně zkontrolovat, zda výpočet správně vyhodnocuje polohu všech 42 objemů je obtížné. Proto jsem vytvořil simulační model složený z koulí (Obr. 39), reprezentující náhradní kolizní model robotu a na něm posuzoval správnost výpočtu.



Obr. 39 Simulační model pro testování výpočtu kolizí

K tomuto modelu jsem vytvořil konzolovou aplikaci, která se připojovala k simulaci a ovládala polohu jednotlivých koulí (Obr. 40). V ní jsem zkoušel nastavovat různé kombinace natočení jednotlivých segmentů mechanismu a sledoval, jestli výpočet kolizí funguje.



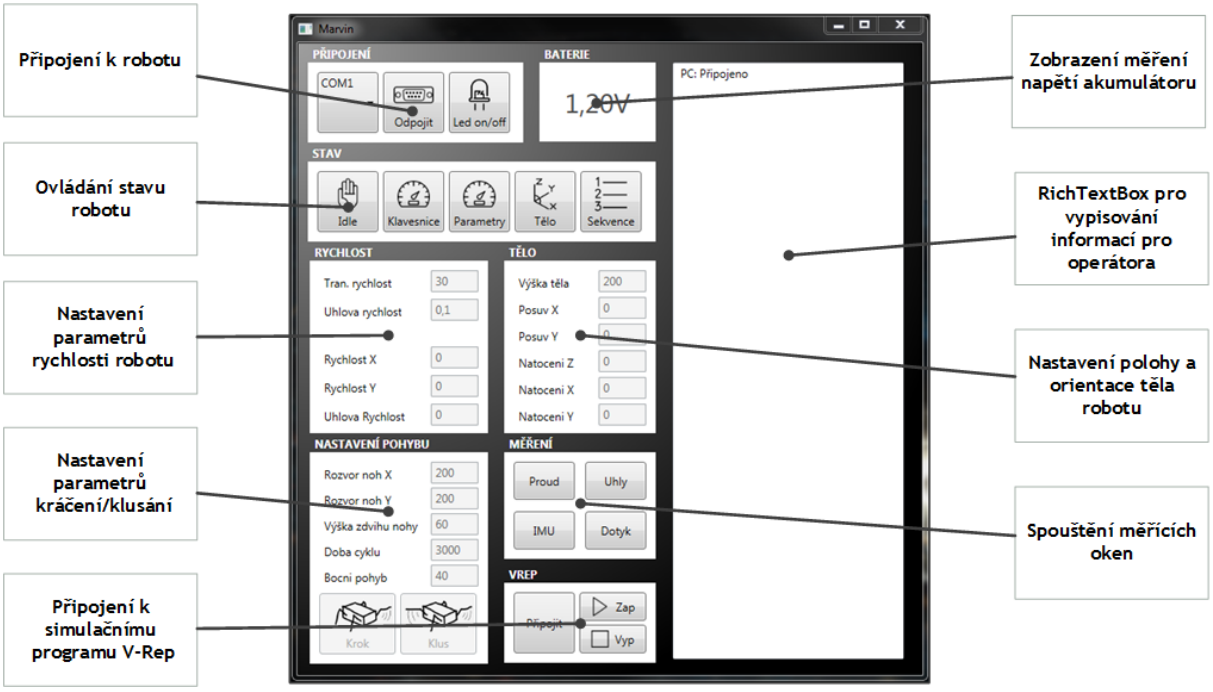
Obr. 40 Aplikace pro vyzkoušení kolizí

6 Softwarová část – operátor

Tako kapitola popisuje aplikaci určenou k ovládání robotu. Aplikace běží na počítači s Windows a k robotu je připojena přes sériový port. Ovládají se přes ni jednotlivé stavy robotu a jejich parametry. Aplikace je může vykreslovat a ukládat data ze senzorů v robotu. Po připojení k programu V-Rep je aplikace schopna ovládat pohyb simulovaného robotu.

6.1 Uživatelské prostředí

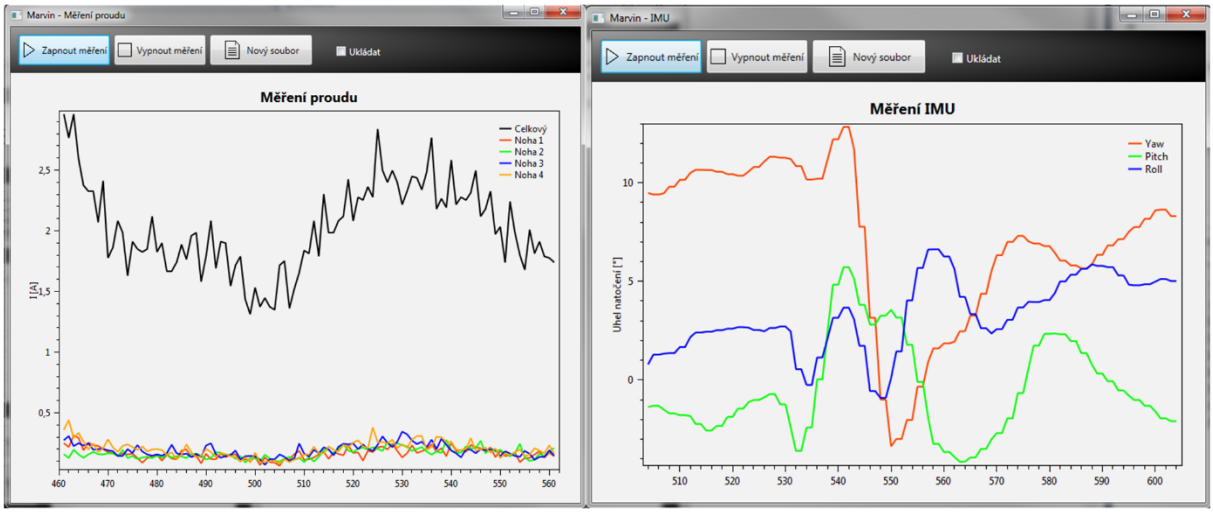
Aplikace je napsaná v .NET Frameworku v jazyce C#. Pro grafické rozhraní je použita knihovna tříd Windows Presentation Foundation, využívající značkovací jazyk XAML. Aplikace obsahuje jedno hlavní okno (Obr. 41), a několik spustitelných oken pro vykreslování měřených dat. Aplikace využívá až 50 MB paměti RAM.



Obr. 41 Hlavní okno aplikace

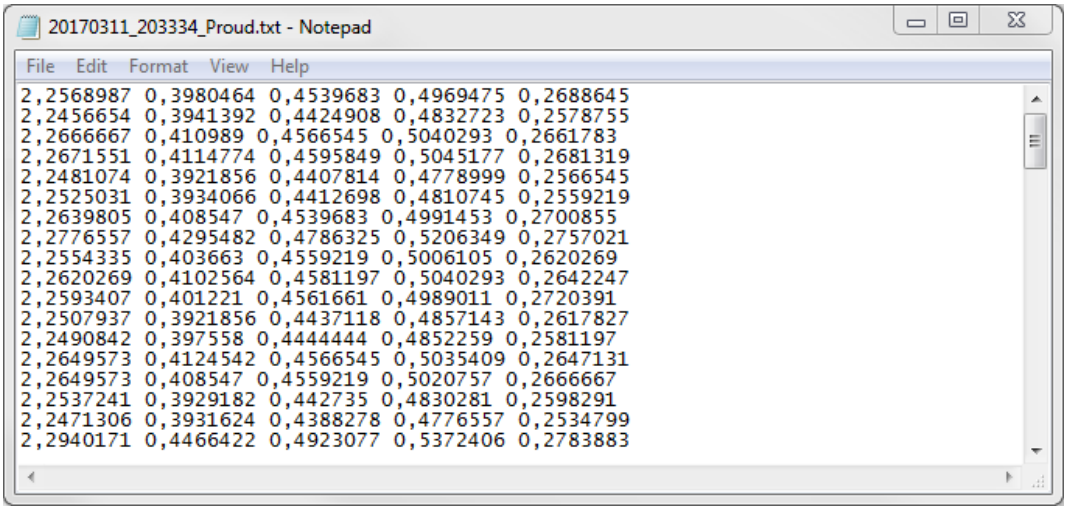
Hlavní okno aplikace obsahuje ovládací prvky pro ovládání sériového portu. Uživatel si z vysouvací nabídky vybere číslo portu a pomocí tlačítka se připojí. Po úspěšném připojení se objeví zpráva v pravém informačním textboxu. Tento textbox zabírá většinu plochy okna a informuje operátora o spouštěných příkazech v PC a v robotu. Po připojení se zpřístupní další ovládací prvky robotu, jako je přepínání stavu robotu. Robot je po zapnutí ve stavu nečinnosti, ze kterého je možno přepnout do kráčení, ovládání pohybu nebo spuštění sekvence. Po přepnutí stavu se aktivují příslušné textboxy, použitelné k ovládání parametrů činnosti robotu. Na hlavním okně se zobrazuje vykreslení pouze jedné měřené hodnoty, napětí akumulátoru, která se v robotu měří a odesílá hned po inicializaci systému. Ostatní měření jsou ve výchozím stavu

vypnuty a jsou spustitelné v jejich příslušných oknech (Obr. 42). Tyto okna pro vykreslování grafů se spouští pomocí tlačítek v hlavním okně.



Obr. 42 Vykreslování dat ze senzorů

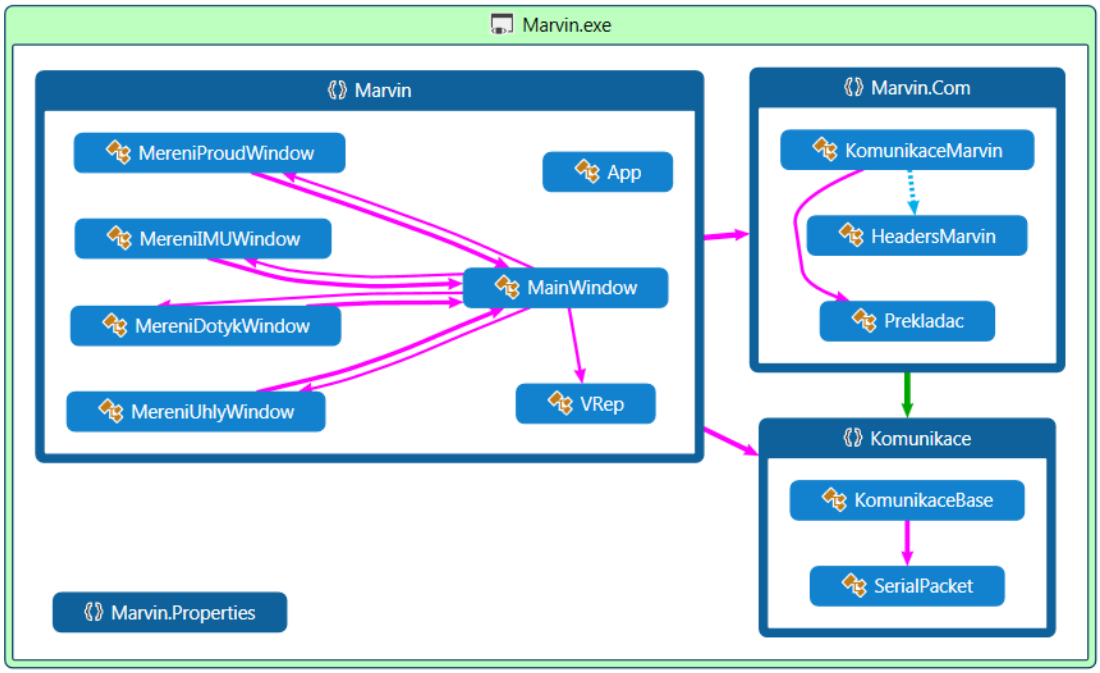
Všechny měřicí okna jsou koncipovány podobně. V horní části okna jsou umístěny tlačítka pro ovládání spouštění a zastavování měření v robotu a tlačítko pro vytvoření nového souboru pro ukládání měřených dat. Většinu plochy okna potom zabírá grafické zobrazení měřených dat. Pro tyto grafy byl použit NuGet balíček OxyPlot.Wpf, který umožňuje vkládání dvourozměrných čarových, bodových a plošných grafů do WPF aplikací. Poté co aplikace na PC přijme paket s měřenými daty se tyto data vloží do grafu a aktualizuje se zobrazení. Zároveň, pokud je vytvořen soubor pro ukládání dat a aktivován checkbox pro ukládání, se data zapisují jako jednotlivé řádky do souboru s označením aktuálního dne a času měření (Obr. 43). Takto uložená data se dají jednoduše importovat do jiných programů jako je například Microsoft Excel nebo MatLab.



Obr. 43 Měření proudu ze dne 11.3.2017

6.2 Struktura aplikace

Aplikace je rozdělena do tří jmenných prostorů (namespace). V namespace Marvin jsou umístěny dialogové okna, jejich ovládací třídy a třída Vrep.cs, určená k připojení k simulačnímu programu.



6.3 Struktura tříd

Většina tříd v programu na PC i v robotu je pro přehlednost rozdělena do regionů s parametry, konstruktory, public metodami a private metodami (Obr. 44).

```
3 references
15  public class KomunikaceBase
16  {
17
18      /// <summary> Událost, spouštěná při úspěšném dekódování paketu
21      public event delSerialPacketReceived onSerialPacketReceived;
22
23      Params
74
75      Constr
91
92      Public
149
150     Private
270 }
```

Obr. 44 Třída rozdělená na regiony Params, Censtr, Public a Private

Výjimkou jsou některé třídy obsahující velké množství metod jako například ovládací třída pro hlavní okno aplikace MainWindow.xaml.cs, ve které jsou metody rozděleny do regionů podle funkce (Obr. 45).

```
15    /// <summary> Interaction logic for MainWindow.xaml
      12 references
18    public partial class MainWindow : Window
19    {
20        Params
58
59        Constr
73
74        Cyklus
242
243       Pripojeni
445
446       Přepínání stavů
539
540       Textboxy
698
699       VRep
733
734       Mereni
759
760       Ostatni
823
824   }
```

Obr. 45 Rozdělení třídy MainWindow.xaml.cs do regionů podle funkce

6.4 Komunikace se simulačním programem

K připojení k simulačnímu programu je použita knihovna pro vzdálený přístup remoteApi.dll [23]. Z této knihovny jsou načítány funkce potřebné pro připojení k V-Rep a ovládání objektů v simulaci.

K této knihovně je napsaná třída Vrep.cs, která je vytvořena specificky pro sestavený model v simulaci. Připojuje se k simulačnímu programu a načítá handle všech dvanácti motorů podle jejich názvu v simulaci. Pomocí těchto načtených identifikačních handle potom nastavuje zadané úhly pohonů na simulovaném robotu.

Protože jsou veškeré výpočty pohybu a kinematiky prováděny v robotu, ovládací program na PC pouze čeká na příchozí paket s hodnotami úhlů pro pohony. Podle příchozích dat spustí metodu pro nastavení motorů ve třídě Vrep.cs a simulace je nastavena. Plynulost pohybu v simulaci tedy závisí na intervalu odesílání dat z robotu do PC (Obr. 46).

Obr. 46 Odesílání dat z robotu do simulace

6.5 Komunikace s robotem

K přenosu dat mezi robotem a operátorem je použit sérový port. Komunikace probíhá ve formě paketů s danou strukturou. Tato struktura je definována ve třídě SerialPacket.cs a má následující strukturu:

Tabulka 3 Struktura paketu

's'	'n'	'p'	Header	Délka	Data 0	Data 1	…	Data n	Checksum HB	Checksum LB

První tři bajty označují začátek paketu, a právě tyto bajty jsou hledány v datovém bufferu po přečtení příchozích dat.

Po těchto bajtech následuje Header bajt, který označuje obsah paketu. Tyto headery jsou definovány ve třídě HeadersMarvin.cs (Obr. 48). Pro správnou funkci programu je tedy důležité, aby obě aplikace, v PC i v robotu, obsahovaly totožnou definici headerů.

Jeden bajt označuje délku datové části paketu. Pokud je tato délka nula, neobsahuje paket žádné data a má tedy minimální možnou délku 7 bajtů. Prázdný paket lze použít například pro spouštění příkazů bez zadaných parametrů.

Poté následuje datová část paketu. Počet bajtů v této části je řízen předchozím bajtem délka. Tato část paketu může být dlouhá až 255 bajtů. Jsou v ní uloženy přenášená data, podle druhu paketu to mohou být parametry rychlosti, měřená data ze senzorů, kontrolní textové řetězce a další informace. Protože se jedná o bajtové pole, data musí být nejprve převedeny z původního datového typu do byte[]. K převodu datových typů byla vytvořena třída Prekladac.cs.

Poslední dva bajty jsou kontrolní suma neboli Int16 číslo, které vznikne sečtením všech předchozích bajtů v paketu (Obr. 47). Tato kontrolní suma se po přijetí paketu vypočítá a porovná s hodnotou uvnitř paketu. Pokud se obě kontrolní sumy shodují, paket byl přečten bezchybně a spustí se daná událost.

```
68          public void ComputeChecksum()
69          {
70              int checksum = 0;
71              checksum += 's';
72              checksum += 'n';
73              checksum += 'p';
74              checksum += (byte)Header;
75              checksum += DataLength;
76              for (int i = 0; i < DataLength; i++)
77              {
78                  checksum += Data[i];
79              }
80              Checksum = checksum;
81          }
```

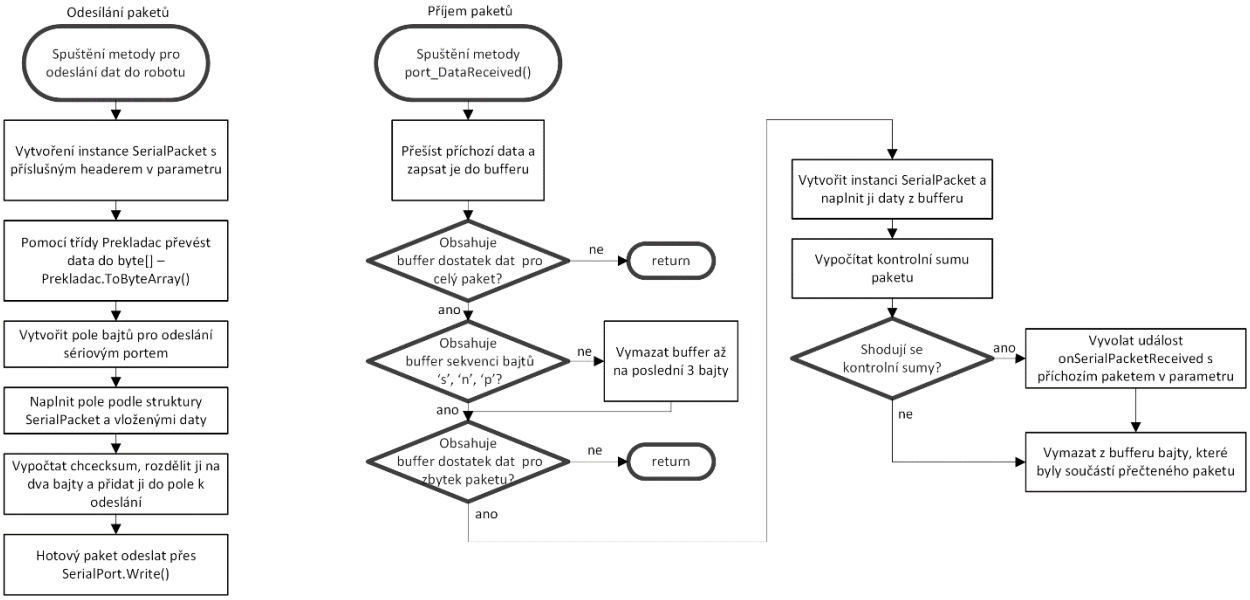
Obr. 47 Výpočet kontrolní sumy

```
27 references
class HeadersMarvin
{
    ////////////////////// Hlavni stavy robotu ////////////////////////////////
    /// <summary> Packet obsahuje prikaz pro prepnuti do necinneho stavu
    public static byte StavIdle = 40;
    /// <summary> Packet obsahuje prikaz ke spusteni pohybu
    public static byte StavPohyb = 41;
    /// <summary> Packet obsahuje prikaz ke spusteni primeho rizeni
    public static byte StavPrime = 42;
    /// <summary> Packet obsahuje informace pro ovladani sekvence
    public static byte StavSekvence = 43;

    ////////////////////// Ovladani parametru ////////////////////////////////
    /// <summary> Packet obsahuje rychlosti pohybu robotu
    public static byte PohybRychlost = 50;
    /// <summary> Packet obsahuje nastaveni pro generator pohybu
    public static byte PohybNastaveni = 51;
    /// <summary> Packet obsahuje uhly natoceni tela robotu a vysku tela robotu
    public static byte TeloOrientace = 52;
    /// <summary> Packet obsahuje prikaz pro prepnuti robotu do rezimu Krokovani
    public static byte Krok = 53;
    /// <summary> Packet obsahuje prikaz pro prepnuti robotu do rezimu Klusani
    public static byte Klus = 54;
    /// <summary> Packet obsahuje vysku tela robotu
    public static byte TeloVyska = 55;
```

```
    ////////////////////// Vedlejsi a doplnkove funkce//////////////////////////
    /// <summary> Packet obsahuje textovou zpravu. Neco jako Debug.Print
    public static byte Message = 70;
    /// <summary> Packet obsahuje informace o prepinani onboard led
    public static byte Led = 71;
    /// <summary> Packet obsahuje příkaz pro nastavení odesílání měřených dat
    public static byte OdesilatProud = 72;
    /// <summary> Packet obsahuje měření proudu
    public static byte Proud = 73;
    /// <summary> Packet obsahuje příkaz pro nastavení odesílání měřených dat
    public static byte OdesilatDotyk = 74;
    /// <summary> Packet obsahuje měření dotykovych senzoru
    public static byte Dotyk = 75;
    /// <summary> Packet obsahuje příkaz pro nastavení odesílání uhlu natoceni motoru
    public static byte OdesilatMotory = 76;
    /// <summary> Packet obsahuje uhly natoceni jednotlivych motoru
    public static byte Motory = 77;
    /// <summary> Packet obsahuje příkaz pro nastavení odesílání IMU dat
    public static byte OdesilatIMU = 78;
    /// <summary> Packet obsahuje IMU data
    public static byte IMU = 79;
    /// <summary> Packet obsahuje příkaz pro nastavení odesílání napětí baterie
    public static byte OdesilatNapeti = 80;
    /// <summary> Packet obsahuje hodnotu napětí
    public static byte Napeti = 81;
```

Obr. 48 Používané headery při komunikaci

Namespace Komunikace obsahuje třídu KomunikaceBase.cs, která se stará o vytvoření instance SerialPort a jeho otevření. Odesílá pakety a při přijmu nových dat na sériový port dekóduje pakety. Postup při odesílání a přijmu paketů je popsán na Obr. 49. Třída napsána tak aby nebyla specifická pro použití výhradně na tomto robotu, ale lze ji využít i pro jiné aplikace spojení přes sériový port.



Obr. 49 Vývojový diagram odesílání a přijjmu paketů

Třídy specifické pro robot Marvin obsahuje namespace Marvin.Com. Třídu KomunikaceBase.cs dědí KomunikaceMarvin.cs, která obsahuje metody pro vytváření a naplnění paketů daty, podle specifického použití.

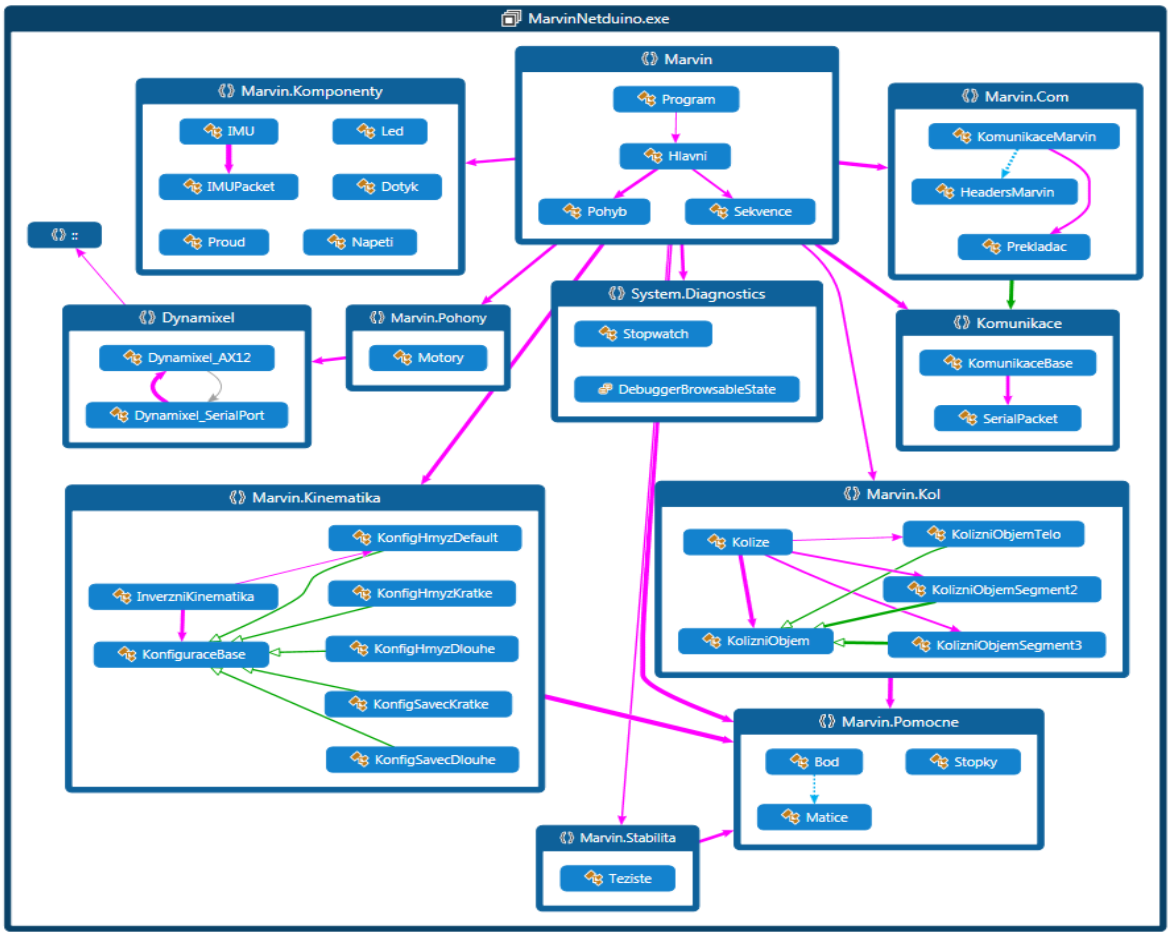
7 Softwarová část – robot

Program v robotu je žízeň příkazy přijímané od operátora. V počátku byly všechny výpočty pohybu a kinematiky počítány na počítači a do robotu se odesílal úhly natočení servomotorů. Přesunutím těchto výpočtů do robotu bylo dosáhnuto větší plynulosti pohybů. Vzhledem k tomu, že je aplikace pro robot programovaná ve stejném jazyce jako aplikace pro operátora, byl tento přesun jednoduchý. K samotnému pohybu tedy robot nepotřebuje být připojen k PC. Program ovládá jednotlivé komponenty a sbírá měřená data ze senzorů, která může následně odeslat operátorovi. Tato kapitola popisuje části programu v robotu, jejich strukturu a použité algoritmy.

7.1 Struktura aplikace

Mikrokontroler Netduino 2 Plus je programován v C# .NET MicroFrameworku v prostředí Visual Studio 2015. Používá 32bitový ARM procesor taktovaný na 168 MHz. Obsahuje 384 kB úložiště pro program a 100 kB RAM [19].

Kód je pro přehlednost rozdělen do jmenných prostorů podle funkce jednotlivých tříd (Obr. 50). Jednotlivé třídy jsou uvnitř navrženy stejně jako v případě aplikace na PC (kapitola 6.3)



Obr. 50 Kódová mapa programu v robotu

7.2 Hlavní část

Nejdůležitější částí programu je třída Hlavni.cs, která se vytváří ihned po zapnutí mikrokontroleru (Obr. 51). Tato třída se stará o správu ostatních částí programu, ovládání cyklu stavu robotu, cyklu odesílání dat do PC a distribuuje data z přijatých paketů. V konstruktoru vytvoří instance všech důležitých částí programu, spustí sekvenci pohybů pro uvedení motorů do výchozí polohy a postavení robotu a spustí vlákno odesílání dat (Obr. 52).

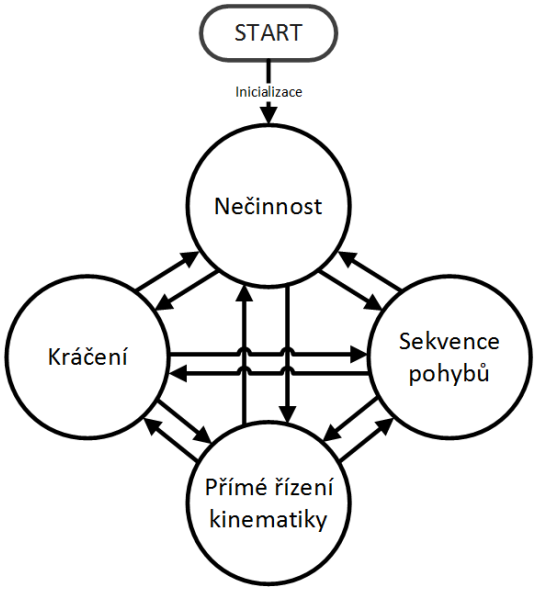
```
13  namespace Marvin
14  {
        0 references
15      public class Program
16      {
            0 references
17          public static void Main()
18          {
19              Debug.Print("Marvin");
20              Hlavni h = new Hlavni();
21              Thread.Sleep(Timeout.Infinite);
22          }
23      }
24  }
```

Obr. 51 Program.cs

```
        1 reference
69      public Hlavni()
70      {
71          _komunikace = KomunikaceMarvin.Instance();
72          _komunikace.onSerialPacketReceived += PacketHandler;
73          _pohyb = new Pohyb();
74          _motory = new Motory();
75          _kinematika = new InverzniKinematika();
76          _sekvence = new Sekvence(_kinematika, _motory);
77          _imu = new IMU("COM4");
78          _kolize = new Kolize();
79          _teziste = new Teziste();
80
81          _sekvence.VychoziPoloha();
82
83          _vlaknoOdesilani = new Thread(new ThreadStart(CyklusOdesilani));
84          _vlaknoOdesilani.Start();
85      }
```

Obr. 52 Konstruktor třídy Hlavni.cs

Činnost robotu je rozdělena na stavy a funguje jako konečný automat [24]. Tedy stroj s konečným počtem stavů, mezi kterými je možno přepínat. Každý stav má svůj cyklus, který jej definuje. Tyto cykly se spouštějí ve vlákně _vlaknoStav. Po inicializaci třídy Hlavni() se robot uvede do nečinného stavu. Poté čeká na příkaz pro přepnutí do jednoho ze tří dalších stavů (Obr. 53). Vzhledem k tomu, že jsou stavy definovány jako metody spouštěné v samostatném vlákně, neznamená nečinný stav úplnou nečinnost robotu. V robotu mohou být zároveň spuštěny jiné procesy jako měření a odesílání dat, běžící ve vlákně _vlaknoOdesilani, které stav robotu neovlivňují. To, že je stav popsán metodou zároveň umožňuje jednoduché přidání dalších možných stavů robotu. V rámci této práce byly vytvořeny čtyři stavy a to: nečinnost, kráčení, přímé ovládání kinematiky a sekvence pohybů.

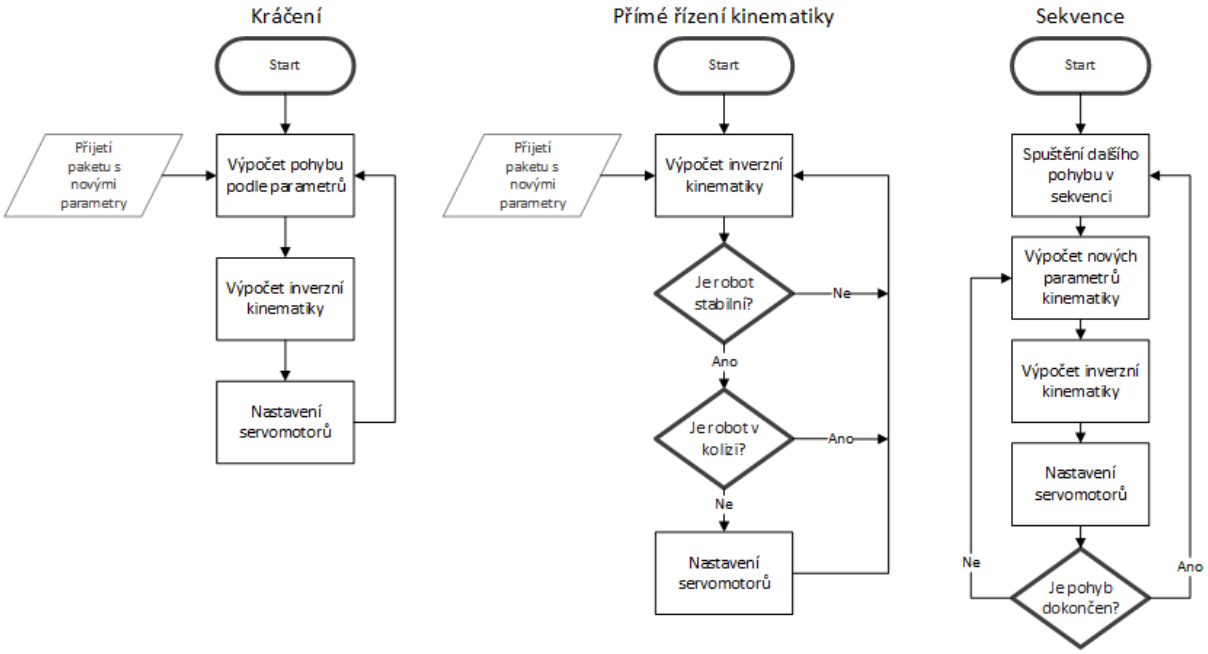


Obr. 53 Schéma konečného automatu v robotu

```
else if (packet.Header == HeadersMarvin.StavPohyb)      // přijetí příkazu pro spuštění cyklu kráčení
{
    if (_vlaknoStav != null && _vlaknoStav.IsAlive)     // pokud je vlákno _vlaknoStav spuštěné, ukončit jej
        _vlaknoStav.Abort();
    _vlaknoStav = new Thread(new ThreadStart(CyklusPohyb)); // vytvořit vlákno _vlaknoStav s metodou pro kráčení
    _vlaknoStav.Start();                                    // spustit vlákno _vlaknoStav
    _komunikace.Message("Robot přepnut do kracejiciho pohybu"); // informovat operátora o změně stavu
}
```

Obr. 54 Přepnutí do stavu kráčení

Přepnutí stavu je vyvoláno přijetím paketu s příslušným příkazem. Nejprve dojde k ukončení vlákna _vlaknoStav a jeho znovuspuštění s jinou metodou (stavem) v parametru (Obr. 54) a zároveň se odešle zpráva o změně pro operátora. Na Obr. 55 je přehled cyklů jednotlivých stavů. Stav kráčení začíná generováním pohybu podle zadaných parametrů, pokračuje výpočtem inverzní kinematiky, která určí úhly natočení pohonů a tyto úhly se poté nastaví. Ve stavu přímého řízení kinematiky stojí robot na místě a lze s ním pohybovat nastavováním parametrů inverzní kinematiky. Pro kontrolu zadaných parametrů je zde implementován výpočet stability a kolizí v mechanismu. Posledním vytvořeným stavem je sekvence. Lze spustit předem definovanou posloupnost pohybů, definovaných jako parametry kinematiky, nebo jako nastavení úhlů jednotlivých kloubů mechanismu. Mezi těmito zadanými polohami robot plynule přechází. Sekvence mají uplatnění pro postavení robotu do výchozí polohy nebo pro jednoduché demonstrace dovedností robotu.



Obr. 55 Vývojové diagramy cyklů jednotlivýc stavů

7.3 Generování pohybu

V této práci je generování pohybu řešeno jako uzavřený systém bez zpětné vazby ze senzorů. Jedná se o parametrický výpočet probíhající v cyklech. Podle parametrů se počítají koncové body noh a poloha těla robotu, s těmito hodnotami poté pracuje inverzní kinematika. Parametry pohybu lze měnit během činnosti robotu. Nastavitelné parametry pohybu jsou:

Tabulka 4 Parametry pohybu

Název parametru	Jednotka	Datový typ	Popis
PeriodaCyklu	ms	float	Perioda jednoho cyklu pohybu robotu. Za tuto dobu uděl každá noha právě jeden krok.
RozvorX	mm	int	Rozvor noh v ose X
RozvorY	mm	int	Rozvor noh v ose Y
Rychlost	mm/s , rad/s	float[]	Rychlost robotu v horizontální rovině. [v_x, v_y, ω]
CasovaniNohy	-	float[]	Určuje kdy v cyklu se dané nohy pohybují
DobaPohybu	-	float	Trvani pohybu jedne nohy. Jako část cyklu
AmplitudaBocni	mm	float	Amplituda se kterou se tělo robotu pohybuje v osách X a Y
VyskaZdvihu	mm	float	Výška, do které jsou zvedány nohy při chůzi

Generátor pohybu pracuje s dvěma poli bodů. Aktuálními polohami a zadanými polohami koncových bodů noh. Při nulové rychlosti jsou tyto polohy totožné. Při zadané nenulové

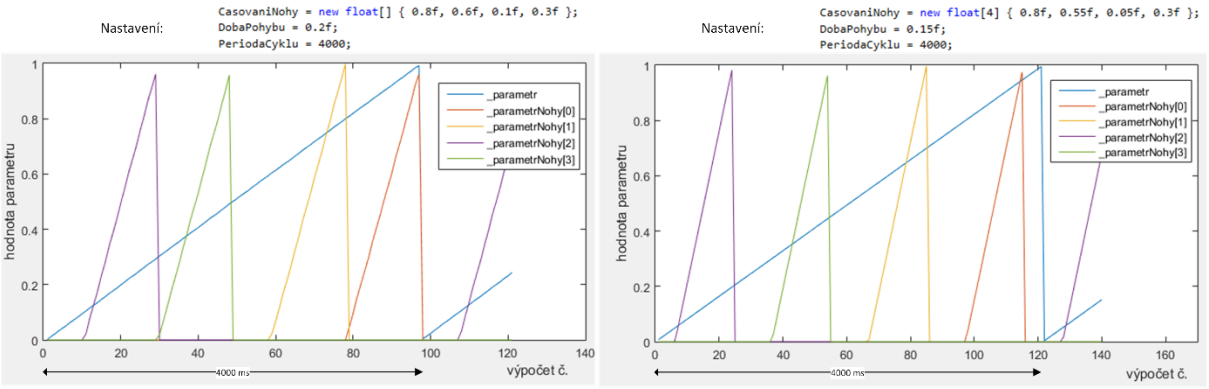
rychlosti se každý výpočetní krok posunou aktuální polohy noh v odpovídajícím směru. Všechny výpočty jsou prováděny v rámci souřadnicového systému S_B, který je umístěný na podlaze.

Řídicí proměnnou ve výpočtu je _parametr. Tato proměnná se během jednoho cyklu mění od 0 k 1, a řídí časování ostatních výpočtů. Je závislá na nastavené periodě cyklu. Výpočtem proměnné _parametr začíná výpočet kráčejícího pohybu. Pokud je zadaná nulová rychlost a žádné nohy nejsou v pohybu, je zbytečné, aby robot zvedal nohy, hodnota proměnné _parametr je nula. Pokud je robot v pohybu, _parametr se vypočítá podle vztahu (3).

```
_parametr = _stopky.ElapsedMilliseconds % PeriodaCyklu / PeriodaCyklu;
```
(3)

```
_parametrNohy[i] = (_parametr - CasovaniNohy[i]) / DobaPohybu;
```
(4)

Časování jednotlivých noh se řídí podle vlastnosti CasovaniNohy. Toto pole má v sobě hodnotu od 0 do 1 pro každou nohu, která určuje kdy v cyklu se daná noha uvede do pohybu. Dobu pohybu jedné nohy řídí vlastnost DobaPohybu a určuje jak dlouho trvá pohyb jedné nohy. Díky této definici lze použít stejný výpočet pro chůzi i klusání. Například při nastavení CasovaniNohy=[0, 0.25, 0.5, 0.75], DobaPohybu=0.2 a PeriodaCyklu=1000, bude druhá noha v pohybu od 250ms do 450ms každý cyklus.



Obr. 56 Časové parametry ve výpočtu pohybu při různých nastavení

Koncový bod nohy opisuje při pohybu půlku elipsy. Tato elipsa je v rovině kolmé k podlaze. V půdorysném pohledu vypadá pohyb koncového bodu jako pohyb po úsečce, výpočet je tedy podle parametrického vyjádření přímky (5). Místo časového parametru t se ve výpočtu objevuje funkce kosinus upravená do oboru hodnot $\langle 0, 1\rangle$ .V ose z se bod pohybuje podle parametrické rovnice elipsy (6). Výpočty pohybu koncového bodu jsou kombinací těchto pohybů (7) (8) (9).

$$x = a_1 + u_1 \cdot t$$
$$y = a_2 + u_2 \cdot t \qquad (5)$$

$$x = a \cdot \cos t$$
$$y = b \cdot \sin t \tag{6}$$

```
_nohyAktualni[i].X = _nohyStare[i].X + (_nohyZadane[i].X - _nohyStare[i].X) *
                     (float)(-Math.Cos(Math.PI * _parametrNohy[i]) + 1) / 2;
```
(7)

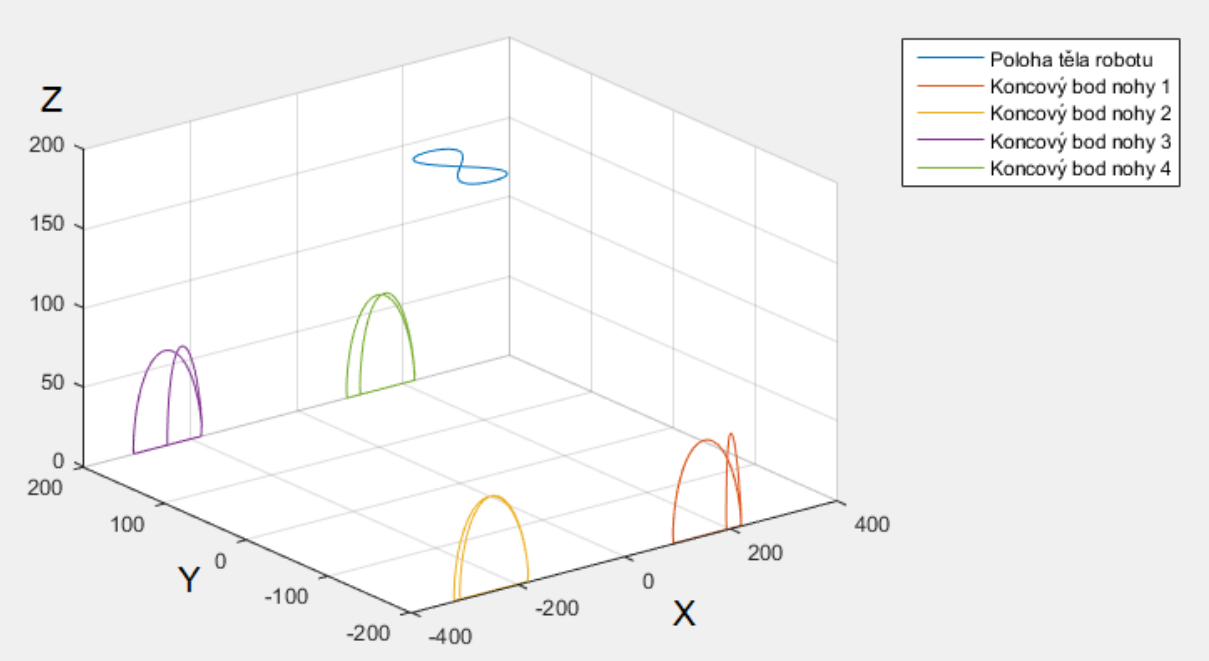
```
_nohyAktualni[i].Y = _nohyStare[i].Y + (_nohyZadane[i].Y - _nohyStare[i].Y) *
                     (float)(-Math.Cos(Math.PI * _parametrNohy[i]) + 1) / 2;
```
(8)

```
_nohyAktualni[i].Z = VyskaZdvihu * (float)Math.Sin(Math.PI * _parametrNohy[i]);
```
(9)

Při statické chůzi jsou vždy alespoň tři nohy v kontaktu s podlahou a stabilita robotu je zajištěna přesouváním těla robotu tak, aby bylo těžiště umístěno nad plochou ohraničenou položenými nohami. Tělo robotu tak v důsledku opisuje tvar čísla 8 [4]. Pohyb těla v osách X a Y je řízen proměnnou _parametr podle vztahů (10).

```
PolohaTela.X = +(float)Math.Sin(_parametr * Math.PI * 4) * AmplitudaBocni;
PolohaTela.Y = -(float)Math.Sin(_parametr * Math.PI * 2) * AmplitudaBocni;
```
(10)

Výstupem generátoru pohybu je tedy pohyb pěti bodů, tj. těla robotu a čtyř noh. Výsledné trajektorie bodů jsou znázorněny na (Obr. 57), které byly zaznamenány při nastavení: výška těla 200mm, rozvory noh 200mm, boční pohyb 40mm, rychlosti x=30mm/s, y=0 a ω=0, perioda cyklu 4000ms a výška zdvihu nohy 60mm.



Obr. 57 Záznam výstupu generátoru pohybu při spuštění kráčení

7.4 Inverzní Kinematika

Inverzní kinematika počítá úhly natočení jednotlivých servomotorů ze zadaných parametrů. Parametry inverzní kinematiky jsou:

Tabulka 5 Parametry inverzní kinematiky

Název parametru	Jednotka	Datový typ	Popis
Noha	mm	Bod[]	Poloha koncových bodů noh
PolohaTela	mm	Bod	Poloha těla robotu v bázovém souřadnicovém systému
Yaw	rad	float	Orientace těla robotu (rotace kolem Z)
Pitch	rad	float	Orientace těla robotu (rotace kolem Y_1)
Roll	rad	float	Orientace těla robotu (rotace kolem X_2)

Pro výpočet inverzní kinematiky mechanismů existuje několik způsobů výpočtu [25]. Mezi ty jednodušší patří algebraické metody, kde je možné vztahy pro jednotlivé stupně volnosti odvodit. Při zvyšování počtu stupňů volnosti tyto metody nestačí a je třeba použít sofistikovanější přístup. Iterační metody řeší inverzní kinematiku použitím sekvence kroků směřujících k postupně lepšímu řešení pro úhly natočení kloubů. Cílem je minimalizovat chybu mezi současnou polohou a zadanou polohou efektoru. Mezi tyto metody lze zařadit inverzi Jakobiho matice, gradientní metody, metoda cyklického sestupu souřadnic, genetické programování a mnoho dalších [26].

Robot Marvin disponuje dvanácti stupni volnosti, je zde však možnost rozdělit inverzní úlohu kinematiky na výpočet čtyř totožných mechanismů o třech stupních volnosti. To umožní použití algebraických metod pro výpočet inverzní kinematiky. Výpočet kinematiky tedy nejprve zjistí polohy koncových bodů noh v souřadnicovém systému dané nohy.

Převod bodu P ze souřadnicového systému S_B do S_R je možno provést vynásobením bodu P_B transformační maticí T_{BR} (11), která se skládá z rotace kolem tří os a translace a popisuje tak orientaci a polohu těla robotu (12). Vzhledem k tomu, že jsou tyto výpočty prováděny na mikrokontroleru v robotu, je vhodné maticovou podobu výpočtu upravit tak, aby nedocházelo ke zbytečnému násobení. V programu MathCad byl odvozen výpočet převedením matice na symbolickou (14). Pro odstranění opakovaných výpočtů goniometrických funkcí jsou zde zavedeny proměnné sr=Sin(roll), cr=Cos(roll), sp=Sin(pitch), cp=Cos(pitch), sy=Sin(yaw) a cy=Cos(yaw).

$$P_R = T_{BR} \cdot P_B \tag{11}$$

$$P_R = T_{roll} \cdot T_{pitch} \cdot T_{yaw} \cdot T_{translace} \cdot P_B \tag{12}$$

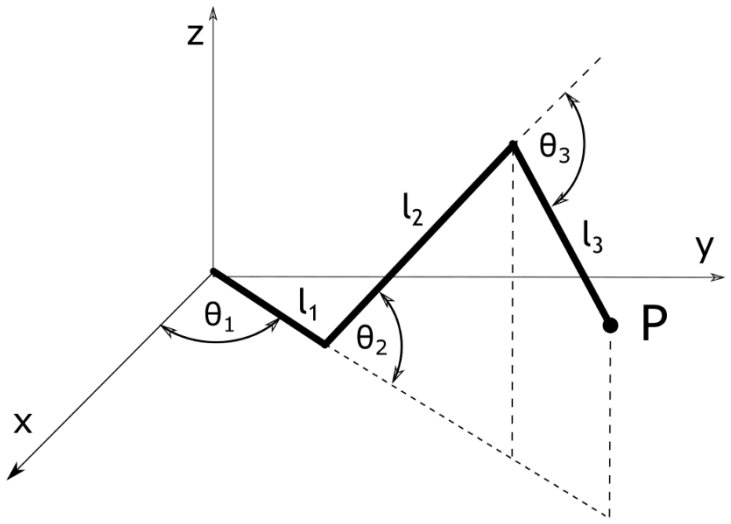
$$P_R := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cr & -sr & 0 \\ 0 & sr & cr & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cp & 0 & sp & 0 \\ 0 & 1 & 0 & 0 \\ -sp & 0 & cp & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cy & -sy & 0 & 0 \\ sy & cy & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & telo_x \\ 0 & 1 & 0 & telo_y \\ 0 & 0 & 1 & telo_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} Px_B \\ Py_B \\ Pz_B \\ 1 \end{pmatrix} \tag{13}$$

$$P_R := \begin{pmatrix} Pz_B \cdot sp + sp \cdot telo_z + Px_B \cdot cp \cdot cy - Py_B \cdot cp \cdot sy + cp \cdot cy \cdot telo_x - cp \cdot sy \cdot telo_y \\ Py_B \cdot cr \cdot cy - Pz_B \cdot cp \cdot sr + Px_B \cdot cr \cdot sy + cr \cdot cy \cdot telo_y - cp \cdot sr \cdot telo_z + cr \cdot sy \cdot telo_x + Px_B \cdot cy \cdot sp \cdot sr - Py_B \cdot sp \cdot sr \cdot sy + cy \cdot sp \cdot sr \cdot telo_x - sp \cdot sr \cdot sy \cdot telo_y \\ Pz_B \cdot cp \cdot cr + Py_B \cdot cy \cdot sr + Px_B \cdot sr \cdot sy + cp \cdot cr \cdot telo_z + cy \cdot sr \cdot telo_y + sr \cdot sy \cdot telo_x - Px_B \cdot cr \cdot cy \cdot sp + Py_B \cdot cr \cdot sp \cdot sy - cr \cdot cy \cdot sp \cdot telo_x + cr \cdot sp \cdot sy \cdot telo_y \\ 1 \end{pmatrix} \tag{14}$$

Poté co jsou polohy koncových bodů noh převedeny do souřadnicového systému robotu, jsou transformovány ještě jednou, tentokrát do souřadnicových systému příslušných noh S_{Ni}. K tomu slouží transformační matice T_{RNi}, které jsou specifické pro aktuální kinematickou konfiguraci mechanismu.

$$P_{N1} = T_{RN1} \cdot P_R \tag{15}$$

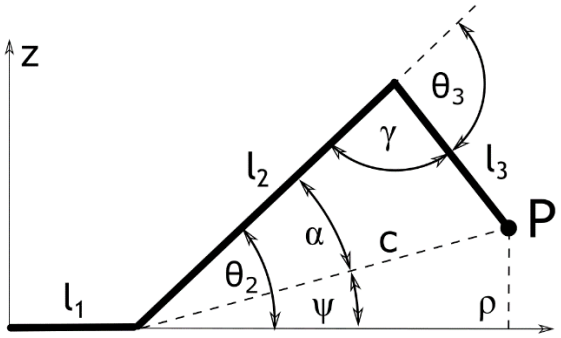
V tuto chvíli je možno řešit inverzní úlohu kinematiky pro mechanismus s třemi stupni volnosti. Ze zadaných souřadnic bodu P v systému S_{Ni} lze pro daný mechanismus odvodit výpočet úhlů θ_1, θ_2 a θ_3. Pro úhel prvního stupně volnosti existuje pouze jedno řešení. Otočení prvního pohonu o úhel θ_1 lze vypočítat podle vztahu (16).



Obr. 58 Inverzní kinematika – θ_1

$$\theta_1 = arctg\left(\frac{x_P}{y_P}\right) \tag{16}$$

Pro zjednodušení výpočtu dalších stupňů volnosti lze výpočet řešit v rovině, procházející osou Z a bodem P (Obr. 59).



Obr. 59 Inverzní kinematika – θ_2 a θ_3

V této rovině se nachází trojúhelník, tvořený délkami druhého a třetího segmentu nohy l_2 a l_3 a vzdáleností c, což je vzdálenost bodu P od osy druhého stupně volnosti. Výpočet nejprve spočítá vzdálenost ρ bodu P od osy Z (17), pomocí této vzdálenosti se spočítá c (18).

$$\rho = \sqrt{x_P^2 + y_P^2} \tag{17}$$

$$c = \sqrt{z_P^2 + (\rho - l_1)^2} \tag{18}$$

Když známe všechny tři strany trojúhelníka, můžeme vypočítat vnitřní úhly α a γ. Použijeme kosinovou větu (19) a (20).

$$\alpha = \operatorname{acos}\left(\frac{c^2 + l_2^2 - l_3^2}{2 \cdot c \cdot l_2}\right) \tag{19}$$

$$\gamma = \operatorname{acos}\left(\frac{l_3^2 + l_2^2 - c^2}{2 \cdot l_3 \cdot l_2}\right) \tag{20}$$

Dále se vypočítá úhel ψ, jako úhel mezi spojnicí osy druhého stupně volnosti s bodem P a rovinou XY.

$$\psi = arctg\left(\frac{z_P^2}{(\rho - l_1)^2}\right) \tag{21}$$

Úhel pro druhý stupeň volnosti θ_2 je tedy součet ψ a α (22). Úhel pro třetí stupeň volnosti θ_3 vyplívá z úhlu γ (23).

$$\theta_2 = \psi + \alpha \tag{22}$$

$$\theta_3 = \gamma - \pi \tag{23}$$

V tuto chvíli jsou vypočítány úhly mezi jednotlivými segmenty nohy jako úhly mezi zvolenými souřadnicovými systémy. Protože servomotory mohou být vůči souřadnicovým systémům úhlově posunuty, nebo mít obrácený smysl otáčení, je do výpočtu zavedena úhlová kompenzace *Epsilon* a výsledné úhly pro servomotory jsou značeny proměnnou *Sigma* (Obr. 60).

```
// uhlova kompenzace ( střední poloha serva neodpovídá nule ve výpočtu, nebo ce servo otáčí jiným směrem )
Sigma[i * 3] = (float)-(Theta[i * 3] + _konfig.Epsilon1); // opacny smysl otaceni
Sigma[i * 3 + 1] = (float)(Theta[i * 3 + 1] + _konfig.Epsilon2);
Sigma[i * 3 + 2] = (float)-(Theta[i * 3 + 2] + _konfig.Epsilon3); // opacny smysl otaceni
```

Obr. 60 Úhlová kompenzace

7.4.1 **Konfigurace kinematiky**

Mechanismus může být sestaven v několika konfiguracích s různě natočenými nohami (kapitola 3.3) a různými délkami segmentů noh (kapitola 3.8). Proto byla vytvořena abstraktní třída KonfiguraceBase.cs, která obsahuje transformační matice pro jednotlivé nohy, délky segmentů nohy a úhlové kompenzace. Pro jednotlivé kinematické konfigurace jsou poté vytvořeny vlastní třídy s nastavenými vlastnostmi, které tuto abstraktní třídu implementují.

Tabulka 6 Vytvořené konfigurace mechanismu

Název třídy	Délka segmentů			Úh. kompenzace		
	l_1	l_2	l_3	ε_1	ε_2	ε_3
KonfigHmyzDefault.cs	49	130	152	0	0.16	1.42
KonfigHmyzDlouhe.cs	49	130	194	0	0.16	1.42
KonfigHmyzKratke.cs	49	83	152	0	0	1.57
KonfigSavecDefault.cs	49	130	152	0	0.16	1.42
KonfigSavecDlouhe.cs	49	130	194	0	0.16	1.42
KonfigSavecKratke.cs	49	83	152	0	0	1.57

```
private Matice[] t = new Matice[4] {
Matice.Translace( -75, -105, 0) * Matice.RotaceKolemZ(Math.PI/2),
Matice.Translace( -75, 105, 0) * Matice.RotaceKolemZ(Math.PI/2),
Matice.Translace( -75, -105, 0) * Matice.RotaceKolemZ(-Math.PI/2),
Matice.Translace( -75, 105, 0) * Matice.RotaceKolemZ(-Math.PI/2), };
```

Obr. 61 Rotační matice T_{RNi} pro konfiguraci hmyz

```
private Matice[] t = new Matice[4] {
Matice.Translace( -11.2f, -40, 120) * Matice.RotaceKolemY(-Math.PI/2) * Matice.RotaceKolemZ(Math.PI),
Matice.Translace( -11.2f, -40, -120) * Matice.RotaceKolemY(-Math.PI/2) * Matice.RotaceKolemZ(Math.PI),
Matice.Translace( -11.2f, 40, -120) * Matice.RotaceKolemY(-Math.PI/2) * Matice.RotaceKolemZ(Math.PI),
Matice.Translace( -11.2f, 44, 120) * Matice.RotaceKolemY(-Math.PI/2) * Matice.RotaceKolemZ(Math.PI) };
```

Obr. 62 Rotační matice T_{RNi} pro konfiguraci savec

7.5 Ovládání pohonů

O ovládání servomotorů Dynamixel se stará třída Motory.cs, která využívá funkcí z Dynamixel_SerialPort.cs.

Při vytvoření instance třídy pro ovládání motorů se třída připojí k jednotlivým servomotorům podle jejich ID 1 až 12. Poté umožňuje nastavovat zadaný úhel nebo číst aktuální úhly natočení jednotlivých pohonů

7.6 Komunikace s IMU

Jednotka UM7 Orientation Sensor používá ke své komunikaci přes sériový port vlastní protokol. Pro tento senzor byla vytvořena třída IMU.cs. Která umožňuje příjem měřených dat ze senzoru a jeho základní nastavení. Pro přístup ke všem nastavitelným parametrům lze využít konfigurační software od výrobce senzoru.

Senzor odesílá data v pravidelném intervalu. Třída IMU v sobě uchovává vždy poslední přijaté hodnoty ze senzoru tak aby k nim byl přístup z jiných částí programu, například ze třídy Hlavni, která tyto data čte a odesílá je operátorovi.

7.7 Měření analogových senzorů

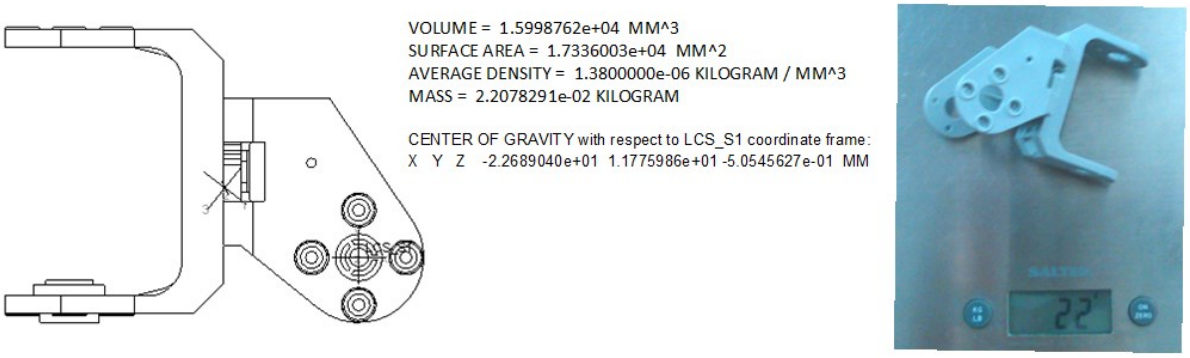
Měření odebíraných proudů a napětí akumulátoru probíhá jednoduše. Jsou pro ně napsané třídy, které obsahují metody pro přečtení aktuálního napětí na příslušném analogovém pinu. Netduino 2 má disponuje 12 bitovým A/D převodníkem. Napětí na analogovém pinu lze měřit metodou ReadRaw(), která vrací celé číslo od 0 do 4095, nebo metodou Read(), která vrací reálné číslo od 0 do 1 a pro toto použití je výhodnější. Tato změřená hodnota se přepočítá na rozsah senzoru, podle vypočítaného vztahu nebo podle výpočtu z datasheetu senzoru. Například výpočet napětí na akumulátoru vychází ze vzorce (2).

```
/// <summary>
/// Napeti na baterii
/// </summary>
1 reference
public static float Zmerit()
{
    return ((float)_a5.Read() * 12.483f);
}
```

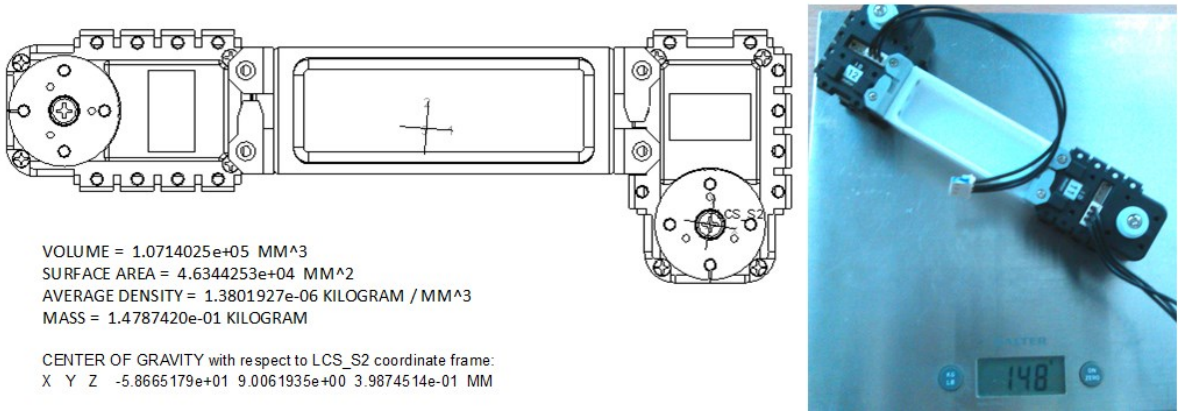
Obr. 63 Měření napětí akumulátoru

7.8 Výpočet polohy těžiště

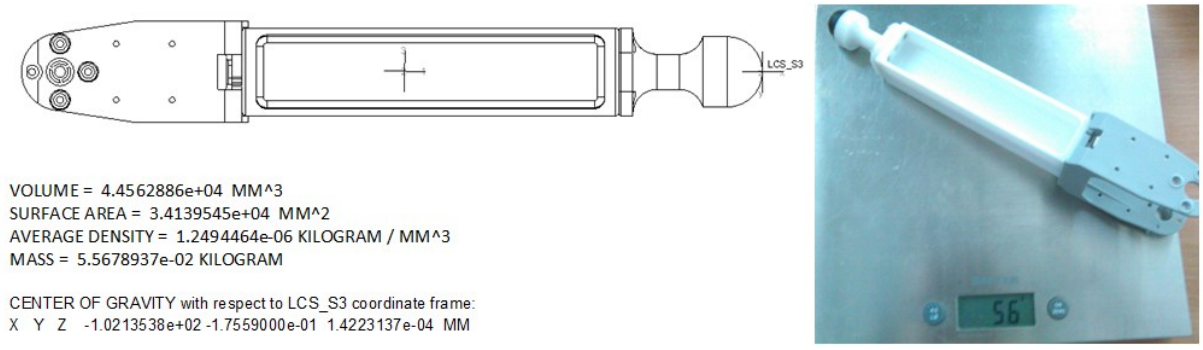
K výpočtu těžiště je potřeba znát polohy těžišť jednotlivých částí mechanismu a jejich hmotnosti. Tyto údaje byly zjištěny z analýzy v Creo Parametric a hmotnosti zkontrolovány postupným zvážením segmentů nohy a těla robotu (Obr. 64,Obr. 65 a Obr. 66).



Obr. 64 Hmotnost a poloha těžiště prvního segmentu nohy

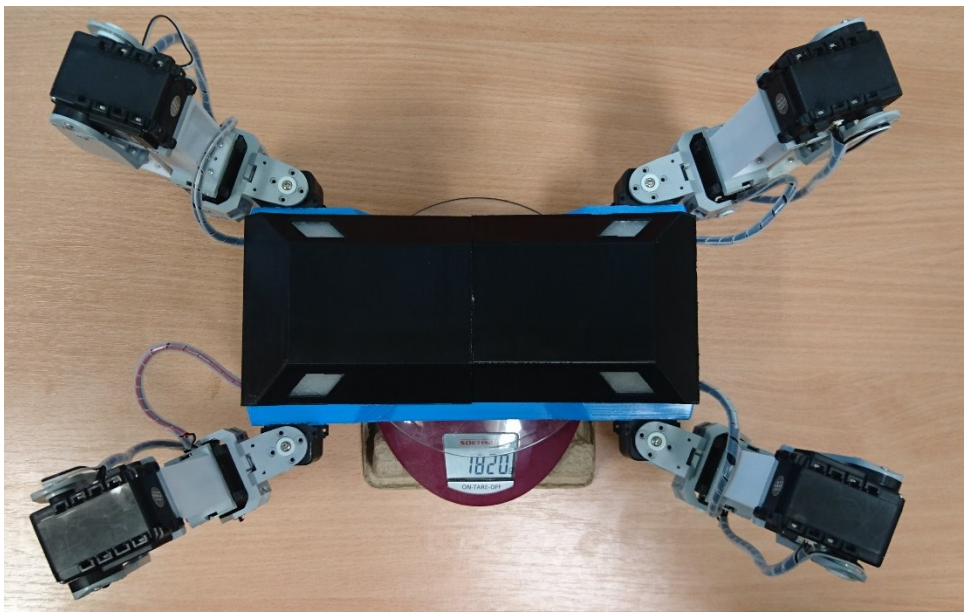


Obr. 65 Hmotnost a poloha těžiště druhého segmentu nohy



Obr. 66 Hmotnost a poloha těžiště třetího segmentu nohy

Hmotnost těla robotu byla při tvorbě výpočtů 1294g a tato hodnota se objevuje ve výpočtech v MathCadu. Výpočet je však napsán tak, aby se dala hmotnost měnit jako parametr. Finální verze robotu váží 1820g (Obr. 67), po odečtení hmotnosti noh má samotné tělo robotu hmotnost 916g.



Obr. 67 Hmotnost finální verze robotu

Polohy těžišť jednotlivých segmentů se vynásobí transformačním maticemi a tím se přepočítají do polohy v souřadnicovém systému jednotlivých noh S_{Ni} .

$$\begin{aligned} P_{t1Ni} &= T_{S1Ni} \cdot P_{t1S1} \\ P_{t2Ni} &= T_{S2Ni} \cdot P_{t2S2} \\ P_{t3Ni} &= T_{S3Ni} \cdot P_{t3S3} \end{aligned} \tag{24}$$

Kde: P_{t1Ni} je poloha těžiště 1. segmentu v SS nohy i, T_{S1Ni} je transformační matice z SS segmentu 1 do SS nohy i, P_{t1S1} je poloha těžiště 1. segmentu v SS 1. segmentu

Tyto transformační matice závisí na proměnných úhlech natočení mechanismu a jejich celý výpočet je uveden v Příloze A. Poloha těžiště jedné nohy se poté vypočítá jako vážený průměr těžišť tří segmentů nohy.

$$P_{tniNi} = \frac{P_{t1Ni} \cdot m_{s1} + P_{t2Ni} \cdot m_{s2} + P_{t3Ni} \cdot m_{s3}}{m_{s1} + m_{s2} + m_{s3}} \tag{25}$$

Kde: P_{tniNi} je poloha těžiště nohy i v jejím SS a m_{s1}, m_{s2} a m_{s3} jsou hmotnosti jednotlivých segmentů.

Tyto těžiště jednotlivých noh se dále přepočítají do souřadnicového systému robotu, pomocí transformačních matic, které popisují kinematickou konfiguraci robotu.

$$P_{tniR} = T_{NiR} \cdot P_{tniNi} \tag{26}$$

Kde: P_{tniR} je poloha těžiště nohy i v SS robotu a T_{NiR} je transformační matice z nohy i do SS robotu.

Výsledné těžiště je vážený průměr těžišť jednotlivých noh a těla robotu.

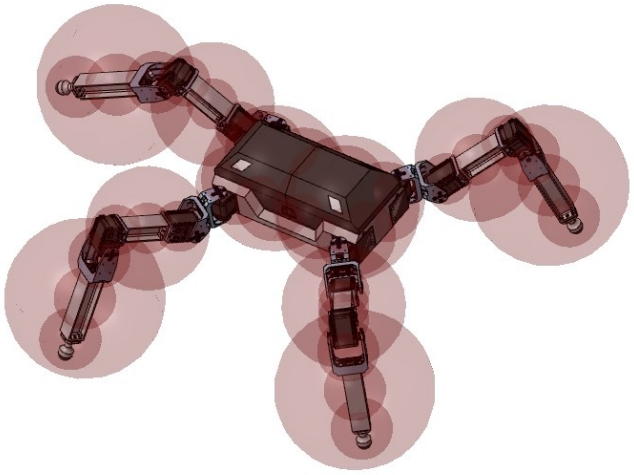
$$P_t = \frac{P_{trR} \cdot m_{tr} + (P_{tn1R} + P_{tn2R} + P_{tn3R} + P_{tn4R}) \cdot m_n}{m_{tr} + 4 \cdot m_n} \quad (27)$$

Kde: P_t je poloha celkového těžiště, P_{trR} je poloha těžiště tela robotu v SS robotu, m_{tr} je hmotnost těla robotu a m_n je hmotnost jedné nohy.

Tento výpočet po implementaci do robotu trval úřibližně 40 ms. Tato doba výpočtu je pro plynulý pohyb robotu příliš velká. Došlo proto k optimalizaci výpočtu těžiště pomocí symbolické matice. Po úpravě trvá výpočet 2,6 ms. Celé odvození symbolické matice v programu MathCad je v Příloze A.

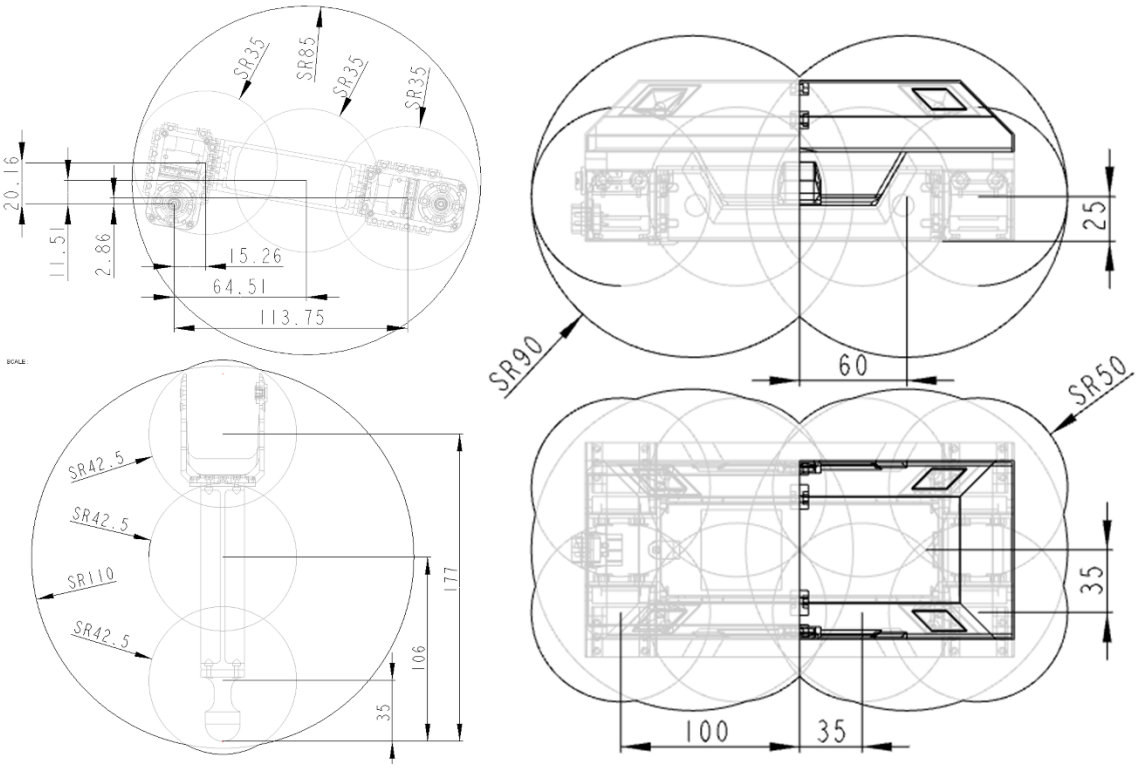
7.9 Výpočet kolizí

Pro detekci kolizí v mechanismu byl model zjednodušen výpočtovým modelem složeným z kulových objemů. Tyto objemy byly umístěny tak aby s malým přesahem pokrývaly jednotlivé části mechanismu ().



Obr. 68 Kolizní model složený z kulových objemů

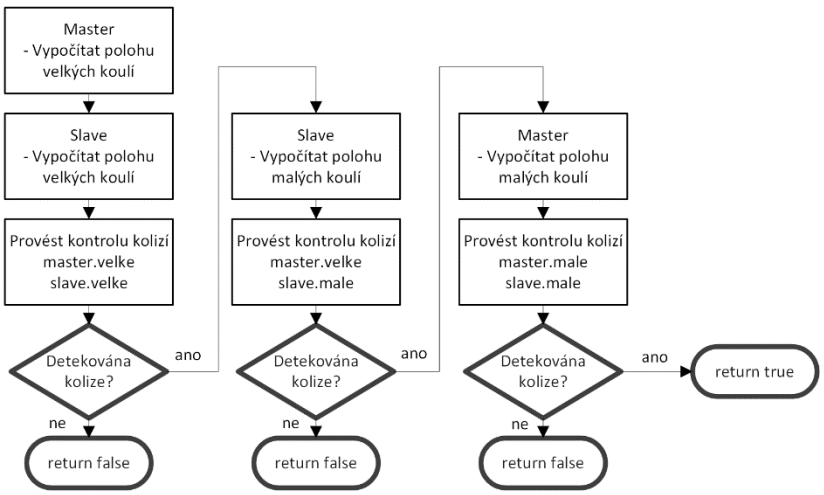
Kolize jsou počítány pro tělo robotu, druhé segmenty a třetí segmenty noh. Aby výpočet nemusel kontrolovat velký počet objemů mezi sebou, má v sobě kolizní model dvě vrstvy objemů, velké a malé. Výpočet tak kontroluje nejprve velké objemy mezi sebou a pokud mezi nimi zjistí kolizi, pokračuje na menší úroveň. Polohy středů kulových objemů a jejich poloměry jsou zobrazeny na Obr. 69.



Obr. 69 Umístění kolizních objemů

Pro výpočet kolizí byla vytvořena třída KolizniObjem. Jedná se o abstraktní třídu, která obsahuje poloměry a polohy malých a velkých koulí v tělese a zadané úhly mechanismu pro výpočet polohy koulí v SS robotu. Tuto třídu dědí třídy KolizniObjemTelo, KolizniObjemSegment2 a KolizniObjemSegment3, které implementují hodnoty pro specifická tělesa a části mechanismu.

Třída KolizniObjem obsahuje metodu pro kontrolu kolize s jinou instancí třídy KolizniObjem. Tato metoda považuje současný objekt (`this`) jako master a objekt v parametru jako slave a v případě kolize snižuje úroveň výpočtu nejprve na slave objektu a poté na masteru. Postup kontroly kolize mezi dvěma částmi mechanismu popisuje Obr. 70.



Obr. 70 Kontrola kolizí – postupné snižování úrovně

Výpočet poloh jednotlivých objemů v tělese je spouštěn až když je potřeba pro porovnání za účelem detekce kolizí a nedochází ke zbytečným výpočtům. Jednotlivé polohy kulových objemů v robotu jsou počítány podobně jako polohy dílčích těžišť při výpočtu celkového těžiště robotu v předchozí kapitole. I v tomto případě bylo provedeno zrychlení výpočtu s použitím symbolické matice. Odvození výpočtů poloh kolizních objemů je v Příloze B.

Kontrola kolizí používá metodu KolizeSlozenychObjemu, která postupně porovná dvě pole bodů, které reprezentují středy koulí jedné úrovně výpočtu.

```
/// <summary>
/// Postupne zkontroluje kolize mezi koulemi v objemech
/// </summary>
/// <param name="master_">Stredy kouli master objemu</param>
/// <param name="slave_">Stredy kouli slave objemu</param>
/// <returns></returns>
3 references
private bool KolizeSlozenychObjemu(Bod[] masterStredy_, float masterPolomer_, Bod[] slaveStredy_, float slavePolomer_)
{
    foreach (Bod m in masterStredy_)
    {
        foreach (Bod s in slaveStredy_)
        {
            bool kontrola = KolizeDvouKouli(m, masterPolomer_, s, slavePolomer_);
            if (kontrola) return true;
        }
    }
    return false;
}
```

Obr. 71 Metoda KolizeSlozenychObjemu

Tím se kulové objemy kontrolovaných těles postupně prověří pomocí metody KolizeDvouKouli, která podle kartézské vzdálenosti středů koulí a jejich poloměru vypočítá, jestli se překrývají nebo ne.

```
/// <summary> Zkontroluje jestli jsou dve koule v kolizi
1 reference
private bool KolizeDvouKouli(Bod stred1_, float polomer1_, Bod stred2_, float polomer2_)
{
    // vzdalenost dvou bodů v prostoru
    //float w_vzdalenost = (float)Math.Sqrt(Math.Pow((stred2_.X - stred1_.X), 2) + Math.Pow((stred2_.Y - stred1_.Y), 2) + Math.Pow((stred2_.Z - stred1_.Z), 2));
    float x = stred2_.X - stred1_.X;
    float y = stred2_.Y - stred1_.Y;
    float z = stred2_.Z - stred1_.Z;
    float w_vzdalenost = (x * x + y * y + z * z);
    // pokud je vzdalenost vetsi nez soucet prumeru, ke kolizi nedochazi
    if (w_vzdalenost > (polomer1_ + polomer2_) * (polomer1_ + polomer2_))
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Obr. 72 Metoda KolizeDvouKouli

Celkový výpočet kolizí poté zastřešuje třída Kolize. Ta obsahuje pole instancí tříd KolizniObjem. Postupně na nich volá kontrolu a poté rozhoduje o kolizním stavu celého mechanismu.

7.10 Sekvence

Přehrávání předem zadaných sekvencí pohybu umožňuje třída Sekvence. Pro definování dílčích pohybů jsou vytvořeny metody:

```
private void Telo(int X_, int Y_, int Z_, float doba_)

private void Rotace(float yaw_, float pitch_, float roll_, float doba_)

private void JednotliveMotory(float[] puvodniUhly_, float[] nastaveneUhly_, float doba_)
```

Metoda Telo() přemístí tělo robotu plynulým pohybem z aktuální polohy do zadané polohy definované v parametrech metody za danou dobu. Metoda Rotace() má podobnou funkci, ale na rozdíl od polohy ovládá orientaci těla robotu. Tyto metody vytváří pohyb mezi aktuálním nastavením kinematiky a zadaným nastavením jako rovnoměrný konstantní pohyb. Při každém výpočtovém kroku posunou nastavení kinematiky o kus k zadané hodnotě tak, aby do ní dorazily v zadanou dobu a po každém takovém posunutí spustí výpočet inverzní kinematiky a nastavení motorů.

Metoda JednotliveMotory() nastavuje přímo hodnoty všech dvanácti stupňů volnosti bez využití inverzní kinematiky. Je použita například v inicializaci a postavení robotu, kdy po zapnutí systému jsou nohy v náhodně otočených polohách a tato metoda je přivede do výchozí polohy všechny ve stejný čas.

Tyto metody se poté mohou skládat za sebe pro vytvoření sekvence pohybů například pro demonstrační účely (Obr. 73).

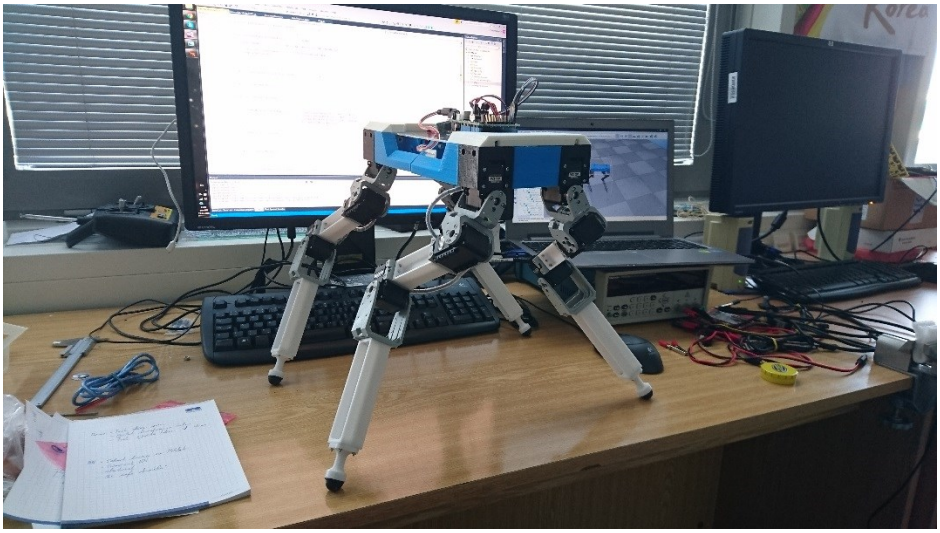
```
/// <summary> Demonstrační sekvence pohybů
0 references
public void Demo()
{
    Debug.Print("Sekvence spustena.");
    while (true)
    {
        //tady nastavit sekvenci pohybů

        Telo(0, 0, 100, 2);
        Telo(0, 0, 200, 2);
        Telo(0, 0, 100, 2);
        Thread.Sleep(1000);

        Telo(0, 0, 100, 0.75f);
        Telo(0, 0, 200, 0.75f);
        Telo(0, 0, 100, 0.75f);
        Telo(0, 0, 200, 0.75f);
        Thread.Sleep(1000);

        Rotace(0.4f, 0, 0, 2);
        Rotace(-0.4f, 0, 0, 4);
        Rotace(0, 0, 0, 2);
        Thread.Sleep(1000);
```

Obr. 73 Začátek demonstrační sekvence pohybů

8 Testování robotu

Testováním fyzického robotu jsem zjišťoval, jak jednotlivé parametry generátoru pohybu ovlivňují maximální rychlost robotu při kráčení a klusání. Postupným upravováním parametrů a zvyšováním rychlosti robot dosáhl maximální rychlosti 160 mm/s při kráčení a 400 mm/s při klusání.

Při testování samotného robotu se objevily některé nesrovnalosti oproti pohybu v simulaci. Největším rozdílem byla vůle v servomotorech. Značné pružení noh se projevilo při konfiguraci savec s dlouhýma nohama (Obr. 74). V této konfiguraci se mohl robot pohybovat pouze velmi pomalu nebo s širokým rozvorem noh.



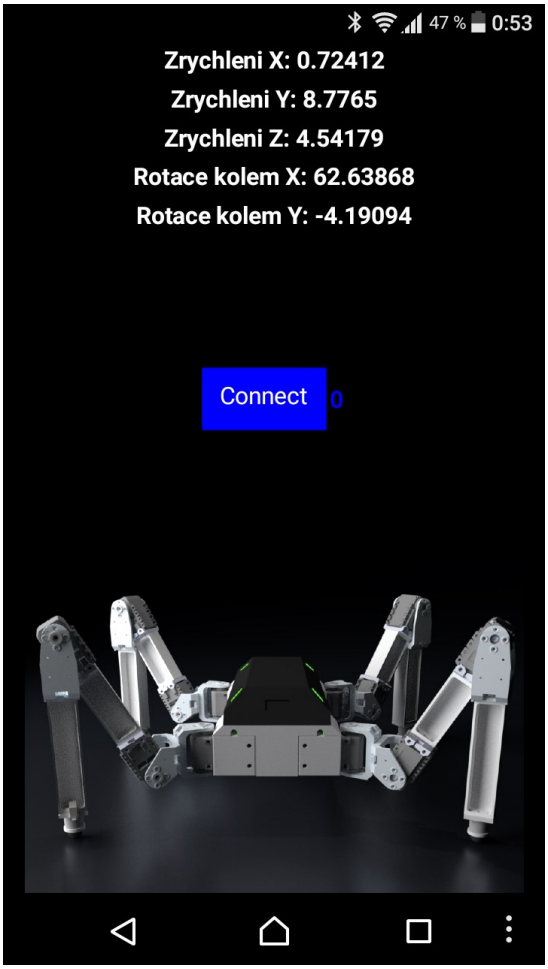
Obr. 74 Test konfigurace savec

Robot byl součástí prezentace Katedry robotiky na dnech NATO 2016 a na dnech otevřených dveří Vysoké školy Báňské – Technické Univerzity Ostrava. (Obr. 75)

Obr. 75 Robot Marvin na dni otevřených dveří VŠB-TUO

S robotem jsem se účastnil soutěže Studentská tvůrčí a odborná činnost (STOČ) na Univerzitě Tomáše Bati ve Zlíně, kde jsem se umístil na prvním místě v kategorii Robotické systémy.

Díky tomu, že jsou všechny výpočty potřebné ke kráčejícímu pohybu prováděny v robotu, lze robot ovládat jakýmkoliv zařízením s Bluetooth konektivitou. V prostředí MIT App Inventor jsem vytvořil aplikaci pro mobilní systém android, která je schopna ovládat parametry robotu náklonem mobilního telefonu s použitím zabudovaného akcelerometru (Obr. 76). Z důvodu přechodu na složitější sériový protokol jsem tuto mobilní aplikaci dále nevyvíjel.



Obr. 76 Testovaná aplikace pro android

Jako další rozšiřování systému se nabízí implementaci sofistikovanějších řídicích algoritmů, které by umožnily robotu pohyb po nerovném povrchu, například po schodech. Tyto algoritmy by využívaly zpětnou vazbu z již zabudovaných senzorů.

Vytvořené simulační modely mohou být použity pro trénování neuronových sítí jako jsou Central pattern generátory, nebo hluboké neuronové sítě. Vytvořené algoritmy mohou být porovnány mezi simulací a realitou.

9 Závěr

Podle specifikovaných požadavků byl navržen čtyřnohý robot se senzorickým subsystémem. Navrhnuté díly byly vytištěny na 3D tiskárně a sestaveny. Návrh robotu prošel několika verzemi, s finální čtvrtou verzí. Mechanismus je navrhnutý tak, aby umožňoval konfiguraci orientace noh a rozměrů článků noh. Pro použitý mikrokontroler byla podle požadavků na systém vyrobena deska, přes kterou se k mikrokontroleru připojují všechny ostatní hardwarové komponenty. Byl vytvořen software pro robot a PC. Tyto dvě aplikace spolu komunikují pomocí vlastního sériového protokolu a navzájem si vyměňují data ohledně parametrů pohybu a kinematiky robotu nebo měření ze senzorů, které lze následně vykreslovat a ukládat na PC. V programu V-Rep bylo vytvořeno několik simulačních modelů, na kterých se otestovaly používané algoritmy. V práci jsou popsány algoritmy generování kráčejícího pohybu, inverzní kinematiky, výpočtu polohy těžiště a detekce kolizí v mechanismu.

Omezujícím prvkem při návrhu algoritmů byla rychlost použitého mikrokontroleru Netduino. Velká část práce se zabývala možnými optimalizacemi algoritmů a jejich zrychlováním tak, aby byl výsledný pohyb robotu plynulý.

Vytvořený systém je navrhnutý tak aby ho bylo možné rozšířit o další funkce, proto je veškerý kód opatřen komentáři a vysvětlivkami. Použité senzory nabízí možnost implementovat zpětnou vazbu do algoritmů kráčení a dovolit tak robotu pohyb ve členitém prostředí.

Citovaná literatura

[1] S. M. S. a. K. J. Waldron, Machine That Walk-The Adaptive Suspension Vehicle, The MIT Press, 1989.

[2] H. Kimura, I. Shimoyam a H. Miura, Dynamics in the dynamic walk of a quadruped robot, Advanced Robotics, 1990.

[3] M. Buehler, R. Playter a M. Raibert, „Robots Step Outside," v *International Symposium on Adaptive Motion of Animals and Machines (AMAM)*, Ilmenau, Germany, 2005.

[4] D. Pongas, M. Mistry a S. Schaal, „A Robust Quadruped Walking Gait for Traversing Rough Terrain," v *Robotics and Automation, 2007 IEEE International Conference on Robotics and Automation*, 2007.

[5] BostonDynamics, „Introducing SpotMini," [Online]. Available: https://www.youtube.com/watch?v=tf7IEVTDjng.

[6] B. Ugurlu, I. Havoutis, C. Semini a D. G. Caldwell, „Dynamic Tort-Walking with the Hydraulic Quadruped Robot-HyQ: Analytical Trajectory Generation and Active Compliance Control," v *IEEE International Conference on Robotics and Automation*, Tokyo, Japan, 2013.

[7] S. K. M. F. M. M. e. a. H. Khan, „Development of the lightweight hydraulic quadruped robot — MiniHyQ," v *Technologies for Practical Robot Applications (TePRA), 2015 IEEE International Conference*, Woburn, MA, USA, 2015.

[8] „The HyQ2Max Robot Can Stand Itself up in Rugged Terrain," technabob, [Online]. Available: https://technabob.com/blog/2016/01/04/hyq2max-walking-robot/.

[9] B. L. J. R. X. R. Yibin Li, „Research of mammal bionic quadruped robots: A review," v *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2011.

[10] F. G. C. Q. X. Z. X. Chen, „Spring parameters design to increase the loading capability of a hydraulic quadruped robot,“ v *Advanced Mechatronic Systems (ICAMechS), 2013 International Conference*, Luoyang, China, 2013.

[11] J. T. K. S. P. J. Cho, „Dynamic walking of JINPOONG on the Uneven terrain,“ v *10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, Korea, 2013.

[12] A. J. Ijspeert, „Central pattern generators for locomotion control in animals and robots: A review,“ *Elsevier,* 2008.

[13] J. S.-V. A. I. S.Gay, „Learning Robot Gait Stability using Neural Networks as Sensory Feedback Function for Central Pattern Generators,“ v *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.

[14] K. e. a. Byl, „Reliable Dynamic Motions for a Stiff Quadruped,“ *Springer,* 2009.

[15] J. G. X. D. H. L. e. a. S. Zhang, „Trot Pattern Generation for Quadruped Robot Based on the ZMP Stability Margin,“ v *International Conference on Complex Medical Engineering*, Beijing, China, 2013.

[16] „Dynamixel AX-12A Robot Actuator,“ TrossenRobotics, [Online]. Available: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx.

[17] „Force-Sensing Resistor: 0.2″-Diameter Circle,“ Pololu Corporation, [Online]. Available: https://www.pololu.com/product/1695.

[18] „Netduino,“ Wikipedia the free encyclopedia, [Online]. Available: https://cs.wikipedia.org/wiki/Netduino.

[19] „Netduino plus 2,“ Wilderness Labs Inc., [Online]. Available: http://www.netduino.com/netduinoplus2/specs.htm.

[20] „UM7 Orientation Sensor,“ redshift labs, [Online]. Available: https://www.redshiftlabs.com.au/sensors/um7-orientation-sensor.

[21] „Turnigy Graphene Professional 5200mAh 3S 15C LiPo Pack w/XT60,“ HobbyKing, [Online]. Available: https://hobbyking.com/en_us/turnigy-graphene-5200mah-3s-15c-w-xt60.html?___store=en_us.

[22] „V-Rep,“ Coppelia Robotics, [Online]. Available: http://www.coppeliarobotics.com/index.html.

[23] „Remote API,“ Coppelia robotics, [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm.

[24] „Konečný automat,“ Wikipedia the free encyclopedia, [Online]. Available: https://cs.wikipedia.org/wiki/Kone%C4%8Dn%C3%BD_automat.

[25] L. Bařinka a R. Berka, „Inverse Kinematics - Basic Methods,“ 2002. [Online]. Available: http://old.cescg.org/CESCG-2002/LBarinka/paper.pdf.

[26] V. Mostýn a V. Krys, *Mechatronika průmyslových robotů,* první editor, 2010, p. 201.

[27] O. K. D. W. e. a. Hyoungkwon K, „Foot trajectory generation of hydraulic quadruped robots on uneven terrain,“ v *Proceedings of the 17th International Federation of Automation Control*, Elsevier, 2008.

Přílohy

Na CD

A. Výpočet těžiště v MathCad
B. Výpočet kolizí v MathCad
C. Zjednodušení výpočtu kinematiky v MathCad
D. Programy pro PC a robot
E. Soubor datasheetů použitých hardwarových komponentů
F. 3D modely sestavy v Creo Parametric
G. STL soubory pro 3D tisk
H. Simulační modely ve V-Rep